PONTIFICIA UNIVERSIDAD CATOLICA DEL
ECUADOR FACULTAD DE INGENIERIA
ESCUELA DE SISTEMAS

ANÁLISIS DE SENTIMIENTOS EN REDES SOCIALES SOBRE EL USO Y AVANCE DE LA INTELIGENCIA ARTIFICIAL

AUTOR

Adam Isaac Cabrera Pasquel

DIRECTOR

Charles Escobar T.

QUITO DM, 2024

I. Índice

II. Índice de imágenes

III. Índice de tablas

1. Introducción

1.1. Justificación

Comprender los pensamientos que la comunidad en línea tiene sobre el uso y avance de la inteligencia artificial es importante para conocer sus preocupaciones e intereses en el tema. Este estudio analizará los comentarios de redes sociales para identificar las emociones y actitudes empleadas en los comentarios, ofreciendo una perspectiva clara sobre las preocupaciones, intereses y prioridades de las personas sobre la Inteligencia Artificial.

1.2. Planteamiento del problema

Actualmente las personas se expresan en línea más que nunca, compartiendo sus opiniones, experiencias y preocupaciones sobre una variedad de temas, incluido el uso y avance de la inteligencia artificial, siendo un tema que ha tomado un gran impacto en el mundo actual. Analizar las emociones y percepciones reflejadas en los comentarios de redes sociales nos ayuda a identificar qué temas generan una mayor carga emocional y cómo varía el uso del lenguaje en función a este tema. Esta comprensión puede aportar información valiosa sobre cómo las personas perciben el uso y la evolución de la inteligencia artificial y cómo estas percepciones evolucionan a lo largo de los años, ofreciendo una visión clara de las actitudes y emociones predominantes en el entorno digital.

De lo anterior se puede identificar el siguiente problema principal:

Se carece de un análisis de sentimientos para comprender las emociones y actitudes expresadas por las personas en redes sociales sobre el uso y avance de la inteligencia artificial.

Y los siguientes problemas secundarios:

- No se conoce la disponibilidad de datos sobre el uso y avance de la inteligencia artificial en redes sociales
- No se ha extraído datos de las redes sociales relacionados con el uso y avance de la inteligencia artificial.
- No se conoce cómo se podrían preprocesar los datos extraídos de las redes sociales relacionados con las opiniones acerca del avance de la inteligencia artificial
- No se conoce que modelo responde con estos datos.
- No se han evaluado modelos de aprendizaje automático con estos datos.

1.3. Objetivo General

Realizar un análisis de sentimientos para comprender las emociones y actitudes de las personas sobre el uso y avance de la inteligencia artificial.

1.4. Objetivos Específicos

- Analizar la disponibilidad de datos sobre el uso y avance de la inteligencia artificial en redes sociales.
- Extraer los datos de la red social seleccionada relacionados con el uso y avance de la inteligencia artificial.
- Preprocesar los datos.
- Modelar los datos.
- Evaluar los modelos.

1.5. Alcance

El proyecto se enfocará en realizar un análisis de sentimientos específicamente en los comentarios de redes sociales relacionados con el uso y avance de la inteligencia artificial. Se recopilarán, procesarán y analizarán volúmenes de datos de texto de varios videos relacionados al tema provenientes de las interacciones de los usuarios en esta plataforma. El objetivo es identificar las tendencias emocionales observadas en la comunidad en relación con el tema de la inteligencia artificial.

Además, se entregará un informe final que resuma los hallazgos obtenidos a través del análisis de sentimientos, proporcionando una visión clara de las emociones y actitudes expresadas en los comentarios de redes sociales sobre la inteligencia artificial.

2. Marco Teórico y Conceptual

2.1. Definición y conceptos básicos del análisis de sentimientos

Para entender el comportamiento de una persona es necesario entender de donde surge ese comportamiento, según Narváez, D. (2015):

> "Los sentimientos están fuertemente vinculados al cerebro, determinan como una persona reacciona ante distintos eventos. Se trata de impulsos sensibles hacia aquello imaginado como un hecho positivo o negativo, es decir, los sentimientos son "mini emociones" que determinan el estado de ánimo de una persona." (Narváez U., 2015)

Un análisis de sentimientos se relaciona con el contenido que publican los usuarios en línea, este contenido nos permite generar información útil. (Medhat, W.; Hassan, A.; Korashy, H., 2014)

> "El Análisis de Sentimientos (SA) o Minería de Opinión (OM) es el estudio computacional de las opiniones, actitudes y emociones de las personas hacia una entidad. La entidad puede representar individuos, eventos o temas. Es más probable que estos temas estén cubiertos por revisiones. Las dos expresiones SA u OM son intercambiables, expresan un significado mutuo." (Medhat, W.; Hassan, A.; Korashy, H., 2014, págs. 1093-1113).

Un análisis de sentimientos nos permite entender las emociones de las personas a través de sus valoraciones, opiniones y actitudes orientadas hacia productos, servicios u organizaciones (Agarwal, Nayak, Mittal, & Patnaik, 2020).

2.2. Redes sociales

Para la extracción de datos se consideró principalmente cuatro redes sociales en las que los usuarios comparten activamente sus opiniones y percepciones sobre la inteligencia artificial.

2.2.1. Twitter

Inicialmente explore Twitter debido a su popularidad y su gran cantidad de usuarios, la principal limitación es que existe un límite de caracteres por publicación lo cual obliga a los usuarios a ser concisos, además Twitter tiene una naturaleza de flujo constante de información lo que me lleva a pensar que los usuarios comparten sus pensamientos de manera espontánea en lugar de opiniones más elaboradas, en cuanto al uso de su API encontré las siguientes limitantes: si quiero obtener grandes volúmenes de información a través de su API tiene un límite establecido de comentarios que puedo extraer en base al tipo de plan de suscripción que se haya pagado, lo que considere como una barrera significativa para el análisis.

2.2.2. YouTube

Luego explore YouTube que a diferencia de otras redes donde solo se suelen presentar textos, imágenes o videos cortos, se destaca cuando se trata de ofrecer contenidos audiovisuales extensos o documentales que permiten a los usuarios explorar el tema de manera más profunda y detallada, considero que este tipo de contenido les da a los espectadores más información y tras captar todo este contenido presentado pueden formar una opinión propia y fundamentada sobre el tema. Debido a esto a mi parecer los comentarios de esta red social son más reflexivos sobre las preocupaciones o expectativas de lo que a los espectadores les agrada y desagrada del tema, lo que enriquece el análisis porque contiene comentarios fundamentados y no solo impresiones inmediatas. Además, YouTube ofrece una API que permite recolectar datos de

los videos de la plataforma sin limitaciones.

2.2.3. Mastodon

Esta red social a diferencia de las anteriores tiene una estructura descentralizada que se divide en varios servidores independientes que se enfocan en diferentes temas cada uno, estos tienen sus propias reglas y restricciones dentro de su comunidad, en cuanto a su API para acceder a los datos no tiene ningún tipo de restricción porque es una plataforma de código abierto pero me encontré con una gran limitante y es que debido a que tiene este tipo de estructura, cuando se busca extraer los datos se necesita seleccionar y conectar varias comunidades sobre el mismo tema con diferentes tokens de acceso, por ejemplo, quiero extraer las publicaciones sobre el tema inteligencia artificial, uso el token de acceso “inteligencia artificial” pero una sola comunidad no me da los datos suficientes así que tengo que conectarme a varias comunidades sobre el tema con diferentes tokens de acceso, esto complica la recolección de datos e incluso no me da un grupo de datos consistentes que sean variados y más completos.

2.2.4. Instagram

Instagram es conocido por ser una plataforma enfocada en el contenido visual pero dentro de estas publicaciones existen descripciones y comentarios sobre la imagen o el video presentado, su API me permite obtener las publicaciones con sus hashtags y comentarios en base a tokens de acceso, la limitante es que solo se pueden hacer ciertas solicitudes por hora y por token de acceso, esto hace que sea necesario recopilar los hashtags que se consideren más relevantes sobre el tema del que se quiera extraer las publicaciones y los comentarios, además de tener que esperar cada hora para extraer nuevos datos, y por lo general tampoco se tiene suficiente contenido textual que se considere relevante para un análisis variado.

La decisión final es usar la red social YouTube como la fuente principal de extracción de datos, no solo por las restricciones de acceso en comparación con otras redes sociales sino también porque consideré que las opiniones tienen mayor variedad y representatividad para un análisis.

2.3. Scraping de datos

Es un proceso que nos permite extraer y almacenar datos de manera extensa y organizada a través de servicios en línea o de interfaces de programación de aplicaciones (Hiremath S, 2024).

Las herramientas como librerías, aplicaciones, complementos del navegador web y otras más que pueden o no ser gratuitas son las que generalmente se usan para realizar el proceso de Scraping de datos (Rosero Correa, 2021) y por lo general estos datos extraídos pueden guardarse en archivos o bases de datos para posteriormente analizarlos.

2.4. Herramientas para el análisis

2.4.1. Google Cloud YouTube API

La librería googleapiclient.discovery es una herramienta de Python que se utiliza para

realizar consultas que nos permiten interactuar con los servicios de Google como Drive, YouTube, en este caso se utilizara para acceder a la información de videos de YouTube. (Google, 2022)

2.4.2. Pandas

Es una librería para manipulación y análisis de datos construida sobre el lenguaje de programación Python (pandas development team, 2024), utiliza estructura de datos como series y dataframes que nos facilita el realizar operaciones como analizar, limpiar y manipular los datos.

2.4.3. Scikit-Learn

Es una librería de Python que ofrece herramientas para el análisis de datos y la implementación de algoritmos de machine learning. Tiene herramientas para aplicar modelos de regresión, clasificación, también nos permite la evaluación de modelos, preprocesamiento de datos, etc. (Scikit Learn, 2024)

2.4.4. Unicode

Es una librería que convierte los caracteres especiales en el carácter sin el símbolo especial o el acentuado (Python, 2024) para el español, se usa para procesar los comentarios de manera uniforme para el análisis de texto.

2.4.5. NLTK

La librería nltk o Natural Lenguage Toolkit se usa para el procesamiento de lenguaje natural, nos permite usar herramientas para estructurar, analizar y clasificar textos (NLTK, 2024).

2.4.6. Stopwords (NLTK)

Es un módulo de la librería NLTK que nos da una lista de palabras comunes que no aportan significado en el análisis del texto, palabras en el caso del español como el, y, en y demás (NLTK, 2024), para este caso se excluyen dichas palabras durante la creación

de las nubes de texto porque son consideradas irrelevantes para la visualización.

2.4.7. Matplotlib.pyplot

Nos permite crear gráficos de varios tipos y personalizar sus características como las etiquetas, estilos del texto, colores y demás. Se utiliza para visualizar y analizar los datos (Matplotlib, 2024).

2.4.8. Pysentimiento – create_analyzer

La herramienta create_analyzer viene de la librería pysentimiento que se enfoca en clasificar textos para análisis de sentimientos, esta librería tiene varias funciones para otros tipos de análisis relacionados con NLP (Pérez, 2021).

2.4.9. TfidfVectorizer – sckit-learn

Nos permite transformar el texto en vectores numéricos utilizando la técnica de TfidfVectorizer que mide el peso y determina la importancia de las palabras del texto que se piensa analizar (Scikit-learn, 2024).

2.4.10. SMOTE – imbalanced learn

Es un algoritmo de la Liberia imbalanced learn que se usa en modelos desbalanceados porque genera nuevas muestras sintéticas que imitan respuestas reales para las clases minoritarias, esto permite igualar las clases eliminando el desbalance y por ende incrementa el rendimiento de los modelos (imbalanced-learn, 2024).

2.4.11. WordCloud

Es una librería que nos permite generar nubes de palabras para entender la frecuencia de las palabras en los datos, es útil si se quiere encontrar los términos clave del texto (Müller, 2020).

2.4.12. Seaborn

Es una librería para visualizar datos estadísticos, fue construida sobre matploplib por lo que tienen varias funciones de compatibilidad. Ofrece gráficos como mapas de calor, gráficos de regresión y demás. (Waskom, 2024)

2.4.13. Googletrans

Es una librería que implementa el API del traductor de Google, se puede usar para detectar el lenguaje de un texto y traducirlo (PyPI, 2025).

3. Metodología

3.1. Investigación Cuantitativa

La recolección de comentarios que vienen de la red social seleccionada relacionados a la inteligencia artificial se representara con herramientas y otras técnicas estadísticas para proyectar de manera cuantitativa los resultados obtenidos del análisis, por ejemplo, las métricas estadísticas para comprobar la efectividad de los modelos.

Este método de investigación es ideal para este proyecto porque lo que se busca es

obtener conclusiones basadas en el análisis de la cantidad de comentarios extraídos.

3.2. Metodología técnica

El uso de CRISP-DM y KDD que son metodologías técnicas para el análisis de datos garantiza un proceso organizado y eficiente para el análisis, sin embargo, para este trabajo solo se usan las fases de preprocesamiento de datos, modelado y evaluación debido a que son las más relevantes para cumplir con el objetivo del análisis.

- Preprocesamiento de datos: Limpieza, clasificación y balanceo de las clases.
- Modelado: el entrenamiento de los modelos para clasificar los sentimientos de los comentarios.
- Evaluación: se usarán métricas para medir la efectividad de los modelos y garantizar su calidad.

Las demás fases de ambas metodologías como entendimiento del negocio o implementación no se incluyen debido a que este trabajo no se piensa integrar en un entorno real ni realizar ninguno tipo de análisis extenso de un negocio, por lo que no se necesitan para cumplir con el objetivo propuesto. Así el proyecto se enfoca exclusivamente en obtener conclusiones claras y específicas de la postura que tienen los usuarios hacia la inteligencia artificial.

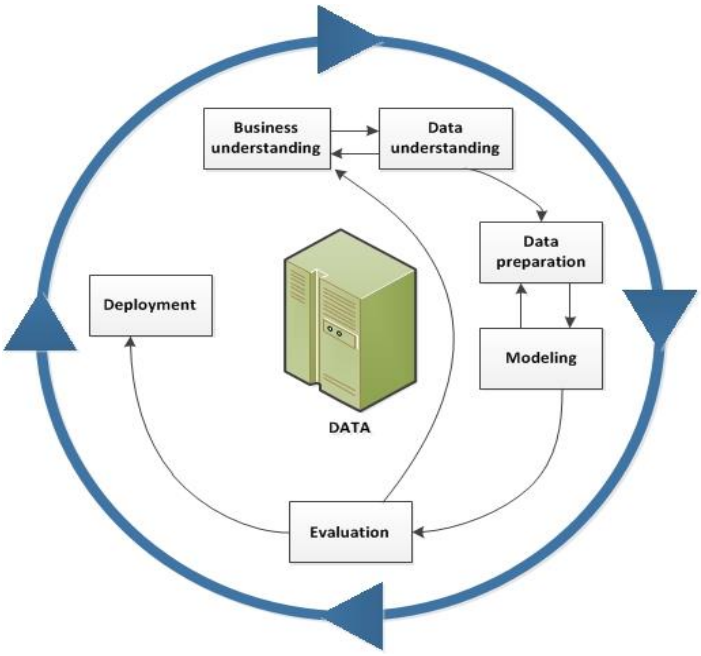

Ilustración 1. Diagrama del proceso CRISP-DM. (IBM - 2021)

4. Análisis de sentimientos para comprender las emociones y actitudes de las personas sobre el uso y avance de la inteligencia artificial.

4.1. Análisis de datos sobre el uso y avance de la inteligencia artificial en redes sociales.

Gran cantidad de información existe gracias a las redes sociales donde se expresan opiniones sobre diversos temas, uno de ellos es la inteligencia artificial, estas plataformas son una fuente valiosa de datos que al ser analizados permiten conocer las actitudes y emociones del cómo se percibe esta tecnología.

El análisis de sentimientos entra dentro del campo del procesamiento del lenguaje natural, este va orientado identificar y clasificar las emociones y actitudes que las personas expresan en texto, según (Isasi, 2021): “El análisis de sentimientos o minería de opinión es utilizado para extraer información sobre la connotación negativa o positiva del lenguaje de un documento.”

Según (Cornejo Urbina, 2022) gracias al gran aumento de información que los usuarios generan cada día los medios han adquirido una relevancia significativa para las organizaciones. Es por eso por lo que este tipo de análisis puede ayudar a responder rápidamente a las opiniones y críticas de los usuarios de un servicio o producto en específico, el escuchar sus sugerencias y aplicarlas no solo mejora el servicio sino también la relación con los clientes (Sharma & Goyal, 2023).

Al evaluar los volúmenes de datos también es posible identificar las tendencias sociales y el cambio constante de las opiniones públicas (Sharma & Goyal, 2023), este proyecto busca encontrar esa postura cambiante que tienen los usuarios de internet sobre el tema de la inteligencia artificial y su constante avance.

4.2. Extracción de datos de la red social seleccionada relacionados con el uso y avance de la inteligencia artificial.

Se realiza una búsqueda de herramientas de Scraping o código para la extracción de los datos, por lo general las redes sociales poseen una API que recopila información del sitio como títulos, recuento de los me gusta, recuento de vistas, comentarios y más. En esta búsqueda se encontró un repositorio de GitHub que indica la estructura para procesar y almacenar de forma eficiente los comentarios de varios videos en un solo archivo haciendo uso de la API YouTube V3 de Google Cloud para extraer los datos respetando los límites de uso y las políticas de la red social.

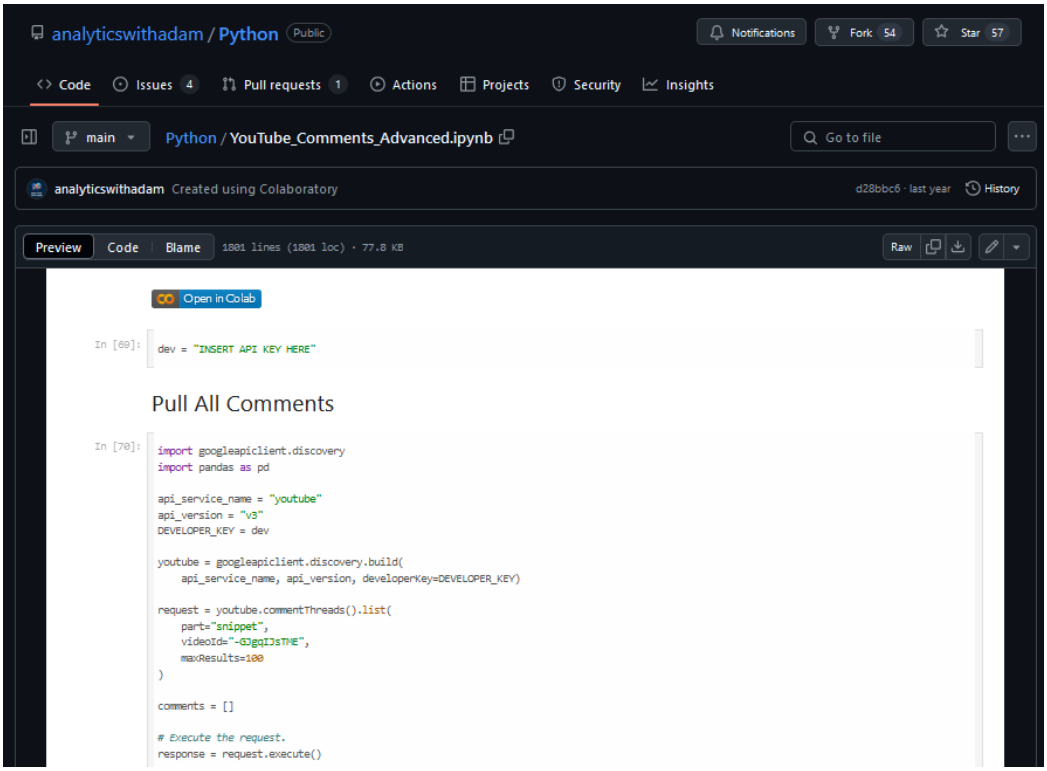

Ilustración 2. Captura del código para extraer comentarios de YouTube (analyticswithadam, 2023).

El único paso adicional fue solicitar una API Key de desarrollador en la plataforma de Google Cloud para acceder a los datos de videos de la plataforma.

4.2.1. Configuración de acceso a los datos a través de la API

Se crea una variable para almacenar la clave de desarrollador de Google Cloud.

```
dev = "AIzaSyD9Jejr1u53TGPfETlgim
```

Ilustración 3. API Key de desarrollador para la plataforma de Google Cloud

Se configura el acceso a la API de YouTube y se inicializa el cliente.

```
import googleapiclient.discovery
import pandas as pd

# Configuración de la API de YouTube
api_service_name = "youtube"
api_version = "v3"
DEVELOPER_KEY = dev  # Clave de desarrollador de la API de YouTube

# Inicializa el cliente para interactuar con la API de YouTube
youtube = googleapiclient.discovery.build(
    api_service_name, api_version, developerKey=DEVELOPER_KEY
)
```

Ilustración 4. Código para configuración de acceso a la API

4.2.2. Función para extraer los comentarios.

La solicitud para obtener los datos de los hilos de comentarios de un video especifico de YouTube se limita a extraer los comentarios en grupos de 100 debido a que la API tiene un sistema de paginación, la solicitud de youtube.commentThreads().list() devuelve los primeros 100 comentarios de ese video, es por eso que más adelante se hace uso de la solicitud nextPageToken para acceder a la siguiente página del video seleccionado.

```
# Función para obtener los comentarios de un video específico
def getcomments(video):
    # Crear una solicitud para obtener los comentarios principales del video
    request = youtube.commentThreads().list(
        part="snippet",       # Especifica que se desea obtener los datos del snippet
        videoId=video,        # ID del video para el cual se obtendrán los comentarios
        maxResults=100        # Número máximo de comentarios a recuperar por solicitud
    )
```

Ilustración 5. Código de la solicitud para extraer los comentarios de un video

Luego se crea una lista vacía para almacenar los comentarios y la variable response

permite ejecutar la solicitud creada para comunicarse con la API.

```
comments = []  # Lista para almacenar los comentarios obtenidos

# Ejecutar la solicitud inicial
response = request.execute()
```

Ilustración 6. Código de la lista vacía y variable para ejecutar la solicitud

Se usa un bucle que extrae los 100 primeros comentarios empezando por los más recientes y los almacena en una lista. La información de la solicitud puede devolver elementos de un comentario como: número de me gusta, texto, id del video, fecha y demás, en este caso solo se extrae el nombre del autor, el texto del comentario y el id del video.

```
# Extraer los comentarios de la respuesta inicial
for item in response['items']:
    comment = item['snippet']['topLevelComment']['snippet']  # Obtener el contenido del comentario
    public = item['snippet']['isPublic']  # Comprobar si el comentario es público
    comments.append([                        # Almacenar los detalles relevantes en la lista
        comment['authorDisplayName'],        # Nombre del autor del comentario
        comment['textOriginal'],             # Texto original del comentario
        comment['videoId'],                  # ID del video asociado al comentario
        public                               # Estado público del comentario
    ])
```

Ilustración 7. Código del bucle para almacenar los comentarios extraídos

Se define un bucle que sigue realizando iteraciones hasta que todas las páginas disponibles del hilo de comentarios de un video sean procesadas, y ya que se tiene la información de respuesta del video (response) se usa la solicitud de nextPageToken para saber si ese video tiene más páginas de comentarios y se crea una nueva solicitud(nextRequest) que incluye el token de las páginas.

```python
# Bucle para manejar comentarios paginados (si hay más de 100 comentarios)
while True:
    try:
        # Obtener el token de la página siguiente si existe
        nextPageToken = response['nextPageToken']
    except KeyError:
        # Si no hay más páginas, salir del bucle
        break

    # Crear una nueva solicitud para obtener la siguiente página de comentarios
    nextRequest = youtube.commentThreads().list(
        part="snippet",       # Especifica que se desea obtener los datos del snippet
        videoId=video,        # ID del video
        maxResults=100,       # Número máximo de comentarios por solicitud
        pageToken=nextPageToken  # Token de la página siguiente
    )
```

Ilustración 8. Código del bucle de la nueva solicitud para obtener las siguientes páginas de comentarios

Otro bucle permite que las solicitudes de las nuevas páginas de comentarios del video sean extraídas y agregadas a la lista existente.

```python
# Ejecutar la solicitud para la siguiente página
response = nextRequest.execute()

# Extraer los comentarios de la respuesta de la página siguiente
for item in response['items']:
    comment = item['snippet']['topLevelComment']['snippet']  # Obtener el contenido del comentario
    public = item['snippet']['isPublic']  # Comprobar si el comentario es público
    comments.append([                     # Almacenar los detalles relevantes en la lista
        comment['authorDisplayName'],     # Nombre del autor del comentario
        comment['textOriginal'],          # Texto original del comentario
        comment['videoId'],               # ID del video asociado al comentario
        public                            # Estado público del comentario
    ])
```

Ilustración 9. Código del bucle anidado para extraer y almacenar todas las páginas de comentarios

Finalmente, se crea un DataFrame con la lista de comentarios extraídos.

```python
# Crear un DataFrame de pandas con los comentarios extraídos
df2 = pd.DataFrame(comments, columns=['author', 'text', 'video_id', 'public'])

return df2
```

Ilustración 10. Código del dataframe para almacenar los comentarios extraídos

4.2.3. Ejecución de la función de extracción

Se utiliza un bucle que usa la función de extracción de comentarios creada anteriormente hasta procesar cada uno de los comentarios de los videos por su Id y los concatena dentro de un solo DataFrame.

```
df = pd.DataFrame() # Dataframe vacío para almacenar los comentarios extraidos

# Bucle para extraer y almacenar los comentarios de cada video por Id en un Dataframe
for i in ['PPMb_rrej5c','MgWtYXcUg9Y','3lnp6mdIf_o','6vMmbqX1pic','gJfp5ryejWA']: # Ids de los videos
  df2 = getcomments(i) # Función para obtener los comentarios de un video específico
  df = pd.concat([df, df2]) # Combina los DataFrames
```

Ilustración 11. Código para la ejecución de la función de extracción

Para tener claro cuantos comentarios han sido extraídos se lleva a cabo un conteo de los comentarios de cada video.

```
# Diccionario que mapea los video_id a los nombres de los videos
video_nombres = {
    'PPMb_rrej5c': '¿Quién mandará en la inteligencia artificial? | DW Documental',
    'MgWtYXcUg9Y': 'Las 3 etapas de la IA, en cuál estamos y por qué muchos piensan que la tercera puede ser fatal',
    '3lnp6mdIf_o': 'La inteligencia artificial, ¿nuestra salvación o condena? | Nosotros y ellos | DW Documental',
    'gJfp5ryejWA': 'El Futuro de la Humanidad | Inteligencia artificial',
    '6vMmbqX1pic': 'Expertos tecnológicos piden una pausa en el desarrollo de la inteligencia artificial'
}

# Obtener los conteos de comentarios por 'video_id'
conteos_video_id = df['video_id'].value_counts().reset_index()
conteos_video_id.columns = ['video_id', 'conteo']

# Agregar la columna de nombres de videos usando el diccionario
conteos_video_id['nombre_del_video'] = conteos_video_id['video_id'].map(video_nombres)

# Mostrar el resultado
conteos_video_id
```

	video_id	conteo	nombre_del_video
0	PPMb_rrej5c	1447	¿Quién mandará en la inteligencia artificial? ...
1	MgWtYXcUg9Y	500	Las 3 etapas de la IA, en cuál estamos y por q...
2	3lnp6mdIf_o	359	La inteligencia artificial, ¿nuestra salvación...
3	6vMmbqX1pic	312	Expertos tecnológicos piden una pausa en el de...
4	gJfp5ryejWA	195	El Futuro de la Humanidad | Inteligencia artif...

```
df.shape
```

```
(2813, 4)
```

Ilustración 12. Código del conteo de comentarios por video

Como resultado se obtuvo un total de 2813 comentarios con las columnas antes designadas de: autor, comentario, el id del video y si su estado es público.

4.3. Preprocesamiento de datos

4.3.1. Limpieza de datos

Dado que se tienen columnas innecesarias en los comentarios extraídos que no son de utilidad para el análisis, se eliminan y se crea un nuevo dataframe solo con la columna de comentarios. Se filtra la columna que se busca conservar que en este caso es “text” y se renombra la columna a “comentarios”.

```
# Filtrar solo las columnas de 'text' (comentarios)
df_esp = df_esp[['text']]

# Renombrar la columna 'text' a 'comentarios'
df_esp = df_esp.rename(columns={'text': 'comentarios'})
```

Ilustración 13. Código para filtrar la columna de interés y renombrar la columna a comentarios

Luego se eliminan los duplicados en caso de existir y se reinicia el índice del nuevo dataframe para evitar futuros errores y reorganizar las filas. Con el fin de comprobar los cambios realizados se imprimen las dimensiones del dataframe antes y después de eliminarlos.

```
df_esp.shape
```

```
(2813, 1)
```

```
df_esp = df_esp.drop_duplicates(subset='comentarios')
```

```
# Se resetea el índice del nuevo DataFrame
df_esp.reset_index(drop=True, inplace=True)
```

```
df_esp.shape
```

```
(2803, 1)
```

Ilustración 14. Código para eliminar comentarios duplicados

Como resultado se sabe que de los 2803 comentarios existen 10 comentarios duplicados. Se crea una función para la limpieza de los comentarios que usa la librería Unicode como

parte del proceso para convertir los caracteres con tilde al carácter equivalente sin tildes, además de procesar los caracteres especiales, las menciones, los dominios y demás.

```python
def limpiar_comentarios(texto):
    texto = unidecode(str(texto))  # Convertir a texto y eliminar tildes
    # Eliminar URLs
    texto = re.sub(r'https?:\/\/.*[\r\n]*', '', texto)
    # Eliminar hashtags y menciones
    texto = re.sub(r'#', '', texto)
    texto = re.sub(r'@[A-Za-z0-9_]+', '', texto)
    # Eliminar dominios web
    texto = re.sub(r'http\S+|www.\S+|pic.\S+|\S+.com\S*|\S+.org\S*|\S+.net\S*', '', texto)
    # Eliminar caracteres especiales, emojis y números
    texto = re.sub(r'[^\w\s]', '', texto)  # Eliminar caracteres especiales
    texto = re.sub(r'[^\x00-\x7F]+', '', texto)  # Eliminar emojis
    texto = re.sub(r'\d+', '', texto)  # Eliminar números
    texto = texto.lower().strip()  # Convertir a minúsculas y eliminar espacios extra
    return texto
```

Ilustración 15. Código de la función de limpieza y procesamiento de caracteres

También se creó una función para eliminar de los comentarios los tiempos o fechas que los usuarios referencian del video para hablar de algo en específico o simplemente comentarios que solo contienen tiempos relacionados al video.

```python
def es_numerico_o_fecha_o_tiempo(texto):
    patron_tiempo = r'^(\s*\d+[:.]\d+[:.]?\d*\s*)+$'  # Detecta patrones como "1:06", "1:06:25", "03:25"
    patron_fecha_o_numero = r'^(\d{1,2}[/-]\d{1,2}[/-]\d{2,4}|\d{4}|\d{1,2} de \w+ de \d{4}|\d+)$'
    return bool(re.match(patron_tiempo, str(texto).strip())) or bool(re.match(patron_fecha_o_numero, str(
```

Ilustración 16. Código de la función para eliminar tiempos y fechas de los comentarios

Gracias a una revisión de los comentarios se supo que existía en los videos algunos comentarios eran creados por inteligencia artificial, en base a su estructura general se creó una función para eliminarlos porque no son relevantes para el análisis.

```python
def es_comentario_ia(texto):
    texto = str(texto).lower().strip()
    # Detectar patrones comunes relacionados con IA en cualquier parte del comentario
    patrones_ia = [
        r'\d{2}:\d{2} \*.*\*',                # Marcadores de tiempo seguidos de descripciones
        r'\b(made with .*ai|powered by ai)\b', # Indicación de que algo fue generado por IA
        r'\bthis text was generated by ai\b',  # Frases indicando generación automática
        r'\b(created|generated|produced) by ai\b',  # Variantes de generación por IA
    ]
    for patron in patrones_ia:
        if re.search(patron, texto):  # Usar search para buscar en cualquier parte del texto
            return True
    return False
```

Ilustración 17. Código de la funcion para eliminar comentarios irrelevantes

Se eliminan los nulos antes y después de ejecutar las funciones creadas, se quitan los

comentarios que tengan menos de 6 caracteres porque no tienen peso en el análisis y por ultimo se visualiza el resultado.

```
# Detectar y eliminar comentarios irrelevantes antes de limpiar
df_esp = df_esp[~df_esp['comentarios'].apply(es_numerico_o_fecha_o_tiempo)]
# Eliminar comentarios generados por IA
df_esp = df_esp[~df_esp['comentarios'].apply(es_comentario_ia)]

# Limpiar la columna `comentarios` después de haber eliminado los irrelevantes
df_esp['comentarios_limpios'] = df_esp['comentarios'].apply(limpiar_comentarios)

# Filtrar los comentarios limpiados con 6 o menos caracteres
df_esp = df_esp[df_esp['comentarios_limpios'].str.len() > 6]

# Ver el DataFrame resultante
df_esp
```

	comentarios	comentarios_limpios
0	La Tecnología esta acabando con los empleos.\n...	la tecnologia esta acabando con los empleos\nd...
1	Esto es más grande que Europa y más grande que...	esto es mas grande que europa y mas grande que...
2	Sanciones a China cuando la tecnología copiada...	sanciones a china cuando la tecnologia copiada...
3	🤔co Colombia no se queda muy atrás de china te...	colombia no se queda muy atras de china tenemo...
4	La biblia dice que el hombre a dominado al hom...	la biblia dice que el hombre a dominado al hom...
...	...	...
2797	es impresionante, me encanta la tecnología 😅 a...	es impresionante me encanta la tecnologia asi...
2798	Excelente información, gracias saludos desde co😊	excelente informacion gracias saludos desde
2799	Sin dudas es el mejor reportaje de divulgación...	sin dudas es el mejor reportaje de divulgacion...
2800	Es tan impresionante que da un poco de miedo \...	es tan impresionante que da un poco de miedo \...
2802	ARIMPRESIONANTE AR	impresionante

2732 rows × 2 columns

Ilustración 18. Código para ejecutar las funciones de limpieza de datos y visualizar los resultados

4.3.2. Traducción de los comentarios

Para utilizar comentarios únicamente del idioma español, se utiliza la librería Translator, que se usa para traducir automáticamente el idioma de un texto. Este enfoque permite eliminar interferencias de otros idiomas que podrían afectar el resultado del análisis.

Se crea una función para traducir comentarios en otros idiomas al español, se analiza si el texto está en español, de ser así simplemente se devuelve el mismo texto sin traducción y en el caso contrario lo traduce.

```python
from googletrans import Translator

# Inicializar el traductor
translator = Translator()

# Crear una función para traducir comentarios
def traducir_comentario(comentario):
    try:
        # Detectar el idioma
        idioma_detectado = translator.detect(comentario).lang
        # Si no está en español, traducir
        if idioma_detectado != 'es':
            comentario_traducido = translator.translate(comentario, src=idioma_detectado, dest='es').text
            return comentario_traducido
        return comentario  # Si ya está en español, devolver el original
    except Exception as e:
        print(f"Error al traducir: {e}")
        return comentario  # Si hay error, devolver el original

# Aplicar la función a la columna "comentarios_limpios"
df_esp['comentarios_limpios_traducidos'] = df_esp['comentarios_limpios'].apply(traducir_comentario)
```

Ilustración 19. Código de la función para traducir el idioma de los comentarios

Por último, se guarda un csv con los comentarios limpios.

```python
#Guardamos el df con los comentarios limpios en un csv
df_limpio.to_csv("comentarios_limpios.csv", index=False, encoding='utf-8')
```

Ilustración 20. Código para guardar los comentarios limpios en un archivo

4.3.3. Etiquetado de comentarios con la librería pysentimiento

Gracias a que dentro de las funciones de la librería de pysentimiento se menciona que las funciones para análisis de sentimientos admiten el idioma español, a diferencia de otras librerías que pueden ser poco efectivas con este idioma, este enfoque especializado permite que se puedan clasificar las emociones dentro de los comentarios con resultados más precisos que otras librerías.

Primero se especifica que se quiere usar la función de análisis de sentimientos y el idioma de los comentarios que se van a clasificar, luego se crea una función que permita clasificar el texto de cada comentario, analiza el contenido del texto y predice su sentimiento además de mostrar la probabilidad de que la clasificación sea correcta en la columna score. Por último, se guardan los resultados de la clasificación en un csv.

```python
# Función para clasificar los sentimientos
def clasificar_sentimiento(comentario):
    if not comentario:
        return 'NEU', 0.0
    resultado = analizador.predict(comentario)
    sentimiento = resultado.output  # 'POS', 'NEG', o 'NEU'
    probabilidad = resultado.probas[sentimiento]  # Probabilidad del sentimiento
    return sentimiento, probabilidad

# Aplicar la clasificación a cada comentario
df[['sentimiento', 'score']] = df['comentarios_limpios'].apply(
    lambda x: pd.Series(clasificar_sentimiento(x))
)

# Guardar resultados en un nuevo archivo CSV
df.to_csv("comentarios_clasificados.csv", index=False)
```

Ilustración 21. Código para clasificar los comentarios con la librería pysentimiento

Para entender mejor el número de comentarios etiquetados por sentimiento, se hace un conteo por categoría y se muestra el resultado tanto en texto como en un gráfico de barras, el conteo en texto para una visualización clara del número exacto de cada categoría, mientras que el grafico de barras identifica visualmente la diferencia entre las cantidades.

```python
# Inicializar contadores
cont_positivos = df[df['sentimiento'] == 'POS'].shape[0]
cont_negativos = df[df['sentimiento'] == 'NEG'].shape[0]
cont_neutros = df[df['sentimiento'] == 'NEU'].shape[0]

# Imprimir los resultados
print(f"Comentarios positivos: {cont_positivos}")
print(f"Comentarios negativos: {cont_negativos}")
print(f"Comentarios neutros: {cont_neutros}")
```

```
Comentarios positivos: 381
Comentarios negativos: 1528
Comentarios neutros: 822
```

Ilustración 22. Código del conteo de categorías (texto)

```python
import matplotlib.pyplot as plt

labels = ['Positivo', 'Negativo', 'Neutro']
colors = ['#456D32', '#6D3030', '#6C6D70']
counts = [cont_positivos, cont_negativos, cont_neutros]

plt.bar(labels, counts, color=colors)
plt.title('Conteo de Sentimientos sobre la IA', fontsize=18)
plt.ylabel('Cantidad')
plt.show()
```

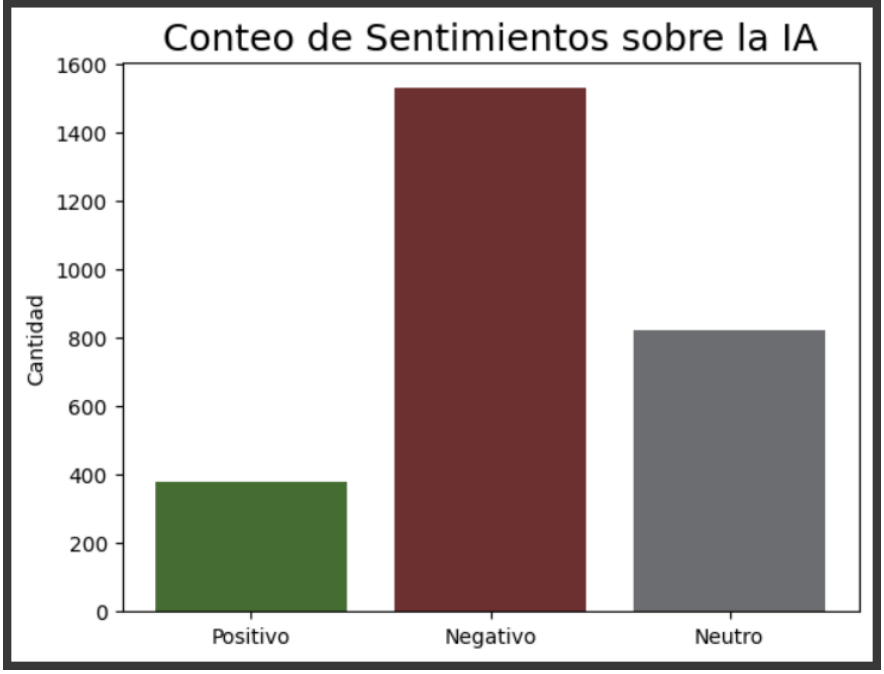

Ilustración 23. Código del conteo de categorías (gráfico de barras)

```
labels = ['Positivo', 'Negativo', 'Neutro']
colors = ['#456D32', '#6D3030', '#6C6D70']
counts = [cont_positivos, cont_negativos, cont_neutros]

plt.figure(figsize=(8, 5))
plt.pie(
    counts,
    labels=[f'{label} ({count})' for label, count in zip(labels, counts)],
    colors=colors, autopct=None, startangle=90, textprops={'fontsize': 12},
    wedgeprops={'edgecolor': 'white', 'linewidth': 1.5}
)

plt.title('Distribución de Sentimientos sobre la IA', fontsize=18)
plt.axis('equal')
plt.show()
```

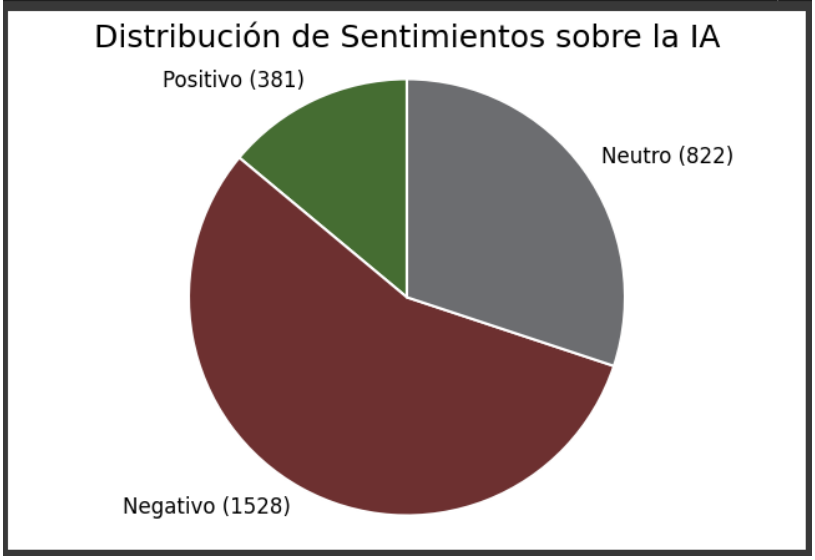

Ilustración 24. Código del conteo de categorías (gráfico de pastel)

4.3.4. Balanceo de comentarios con submuestreo

Antes de balancear los datos es necesario saber cuántos comentarios hay en cada categoría y tomar el número mínimo para igualar la cantidad de cada clase en el submuestreo de los sentimientos.

```python
from sklearn.utils import resample

# Contar los sentimientos
conteo_sentimientos = df['sentimiento'].value_counts()

# Imprimir los resultados
print("Positivos:", conteo_sentimientos.get('POS', 0  ))
print("Negativos:", conteo_sentimientos.get('NEG', 0))
print("Neutros:", conteo_sentimientos.get('NEU', 0))

# Encontrar el número mínimo de ejemplos en las clases
min_count = conteo_sentimientos.min()

# Separar cada clase
positivos = df[df['sentimiento'] == 'POS']
negativos = df[df['sentimiento'] == 'NEG']
neutros = df[df['sentimiento'] == 'NEU']
```

```
Positivos: 381
Negativos: 1528
Neutros: 822
```

Ilustración 25. Código para obtener el número mínimo de ejemplos entre las tres clases

Dado que los comentarios clasificados como positivos tienen el menor número de ejemplos se usa en el submuestreo para igualar las demás categorías, luego se combinan las clases en un solo dataframe y se barajan aleatoriamente.

Por último, se hace un conteo para verificar el balanceo y se guardan los comentarios en un archivo. Las clases terminaron con un total de 390 comentarios cada una.

```python
# Submuestreo de cada clase mayoritaria
positivos_undersampled = resample(positivos, replace=False, n_samples=min_count, random_state=1043)
negativos_undersampled = resample(negativos, replace=False, n_samples=min_count, random_state=1043)
neutros_undersampled = resample(neutros, replace=False, n_samples=min_count, random_state=1043)

# Combinar todas las clases submuestreadas
balanced_comments = pd.concat([positivos_undersampled, negativos_undersampled, neutros_undersampled])

# Barajar el conjunto de datos resultante
balanced_comments = balanced_comments.sample(frac=1).reset_index(drop=True)

# Verificar el balanceo
conteo_balanceado = balanced_comments['sentimiento'].value_counts()
print("Conteo balanceado de sentimientos:")
print(conteo_balanceado)

# Guardar el conjunto de datos balanceado
balanced_comments.to_csv('comentariosBalanceadosUndersamp.csv', index=False)
```

```
Conteo balanceado de sentimientos:
sentimiento
POS    381
NEU    381
NEG    381
Name: count, dtype: int64
```

Ilustración 26. Código del submuestreo para balancear las clases

```
labels = ['Positivo', 'Negativo', 'Neutro']
colors = ['#456D32', '#6D3030', '#6C6D70']
counts = [conteo_balanceado.get('POS', 0), conteo_balanceado.get('NEG', 0), conteo_balanceado.get('NEU', 0)]

plt.bar(labels, counts, color=colors)
plt.title('Conteo de Sentimientos Balanceados', fontsize=18)
plt.ylabel('Cantidad')
plt.show()
```

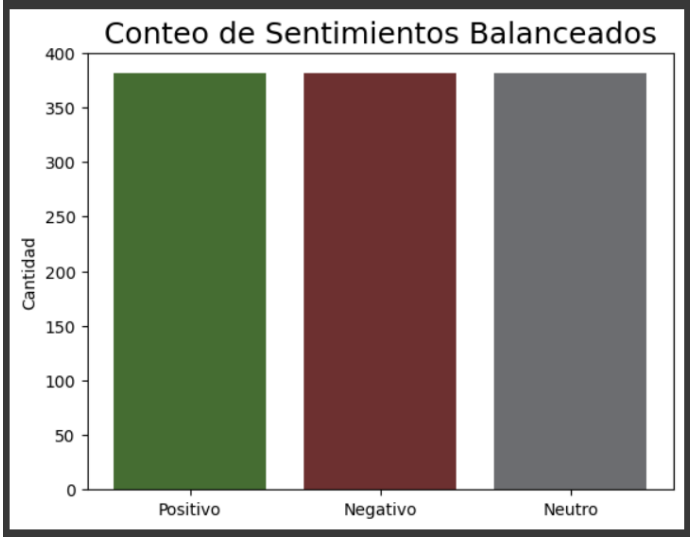

Ilustración 27. Código de visualización del balanceo de clases con submuestreo (gráfico de barras)

```
labels = ['Positivo', 'Negativo', 'Neutro']
colors = ['#456D32', '#6D3030', '#6C6D70']
counts = [conteo_balanceado.get('POS', 0), conteo_balanceado.get('NEG', 0),
          conteo_balanceado.get('NEU', 0)]

plt.figure(figsize=(8, 5))
plt.pie(
    counts,
    labels=[f'{label} ({count})' for label, count in zip(labels, counts)],
    colors=colors, autopct=None, startangle=90, textprops={'fontsize': 12},
    wedgeprops={'edgecolor': 'white', 'linewidth': 1.5}
)

plt.title('Distribución de Sentimientos Balanceados'+ '\n'+'con submuestreo', fontsize=16)
plt.axis('equal')
plt.show()
```

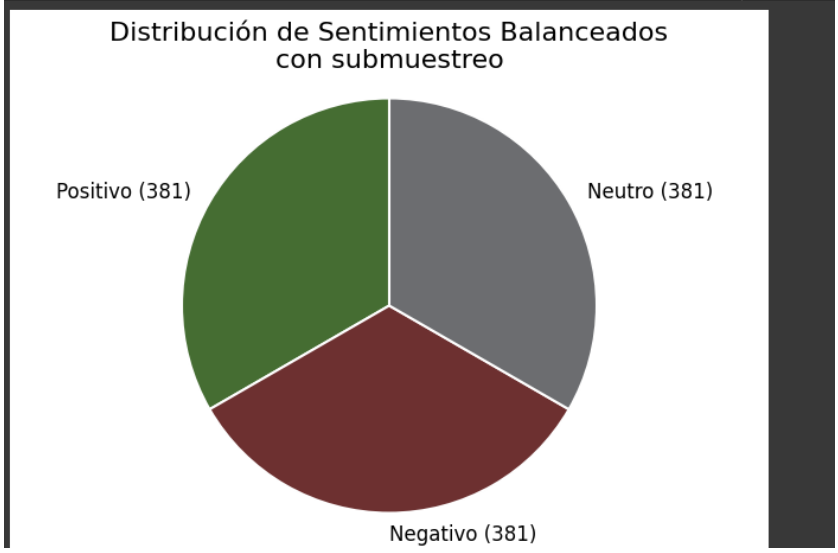

Ilustración 28. Código de visualización del balanceo de clases con submuestreo (gráfico de pastel)

4.3.5. Balanceo de comentarios con sobre muestreo / SMOTE

Esta librería permite crear muestras sintéticas de las clases con menor cantidad de datos e igualarlas a la clase mayoritaria. Gracias al conteo anterior se sabe que los comentarios clasificados como negativos tienen el mayor número de ejemplos, se igualan las demás categorías y se combinan las clases en un solo dataframe y se barajan aleatoriamente.

```
# Vectorizar el texto usando TF-IDF
vectorizador = TfidfVectorizer(max_features=3000)
X = vectorizador.fit_transform(df['comentarios_limpios_traducidos']).toarray()

# Etiquetas (positivo, negativo, neutral)
y = df['sentimiento']

# Aplicar SMOTE
smote = SMOTE(random_state=42)
X_resample, y_resample = smote.fit_resample(X, y)

# Convertimos a DataFrame
df_balanceado = pd.DataFrame(X_resample, columns=vectorizador.get_feature_names_out())
df_balanceado['sentimiento'] = y_resample
```

Ilustración 29. Código del sobre muestreo para balancear las clases con SMOTE

Por último, se hace un conteo para verificar el balanceo y se guardan los comentarios en un archivo. Las clases terminaron con un total de 1514 comentarios cada una.

```
import matplotlib.pyplot as plt

# Conteo de comentarios por sentimiento
conteo_sentimientos = df_balanceado['sentimiento'].value_counts()
# Mostrar el conteo
print(conteo_sentimientos)
```

```
sentimiento
NEG    1528
NEU    1528
POS    1528
Name: count, dtype: int64
```

Ilustración 30. Código del conteo de categorías con sobre muestreo (texto)

```
colores = ['#6D3030', '#6C6D70', '#456D32']
plt.figure(figsize=(8, 6))
bars = plt.bar(conteo_sentimientos.index, conteo_sentimientos.values, color=colores)

plt.title('Balanceo de comentarios con SMOTE (sobre muestreo)', fontsize=16)
plt.xlabel('Sentimiento', fontsize=14)
plt.ylabel('Cantidad de Comentarios', fontsize=14)
plt.xticks(rotation=45)

plt.show()
```

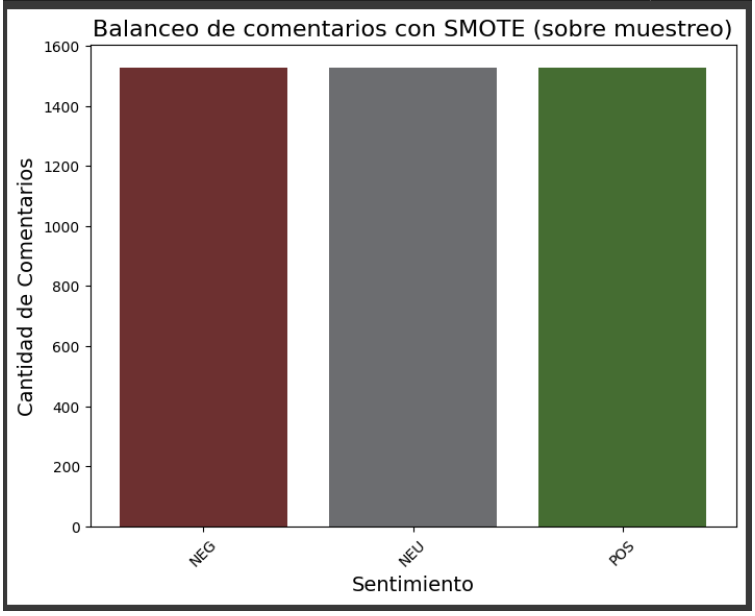

Ilustración 31. Código del conteo de categorías con sobre muestreo (gráfico de barras)

```
labels = conteo_sentimientos.index
counts = conteo_sentimientos.values
colors = ['#6D3030', '#6C6D70', '#456D32']

plt.figure(figsize=(8, 5))
plt.pie(
    counts,
    labels=[f'{label} ({count})' for label, count in zip(labels, counts)],
    colors=colors, autopct=None, startangle=90, textprops={'fontsize': 12},
    wedgeprops={'edgecolor': 'white', 'linewidth': 1.5}
)

plt.title('Balanceo de comentarios con SMOTE (sobre muestreo)', fontsize=16)
plt.axis('equal')
plt.show()
```

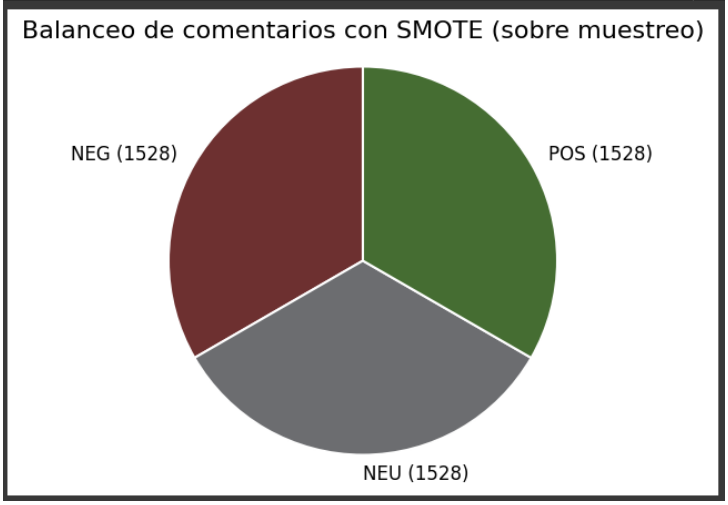

Ilustración 32. Código del conteo de categorías con sobre muestreo (gráfico de pastel)

4.3.6. Nubes de palabras

Las nubes de palabras son de gran ayuda en el análisis porque identifican las palabras clave de un texto y nos dan una visión general de los temas principales dentro de cada categoría.

Nube de palabras de comentarios positivos

Esta nube de palabras muestra y refleja las ideas positivas de los usuarios sobre la inteligencia artificial. Nos da la visión de optimismo por palabras como mejor, excelente, bueno, interesante y demás que demuestran que algunos la ven como una herramienta útil. Otras palabras como crear, desarrollo, trabajo, control y avance demuestran que también se tiene una idea de que la inteligencia artificial puede influir en el progreso y el bienestar de las personas de manera positiva, los comentarios de esta clase muestran que los usuarios tienen un interés positivo sobre el tema.

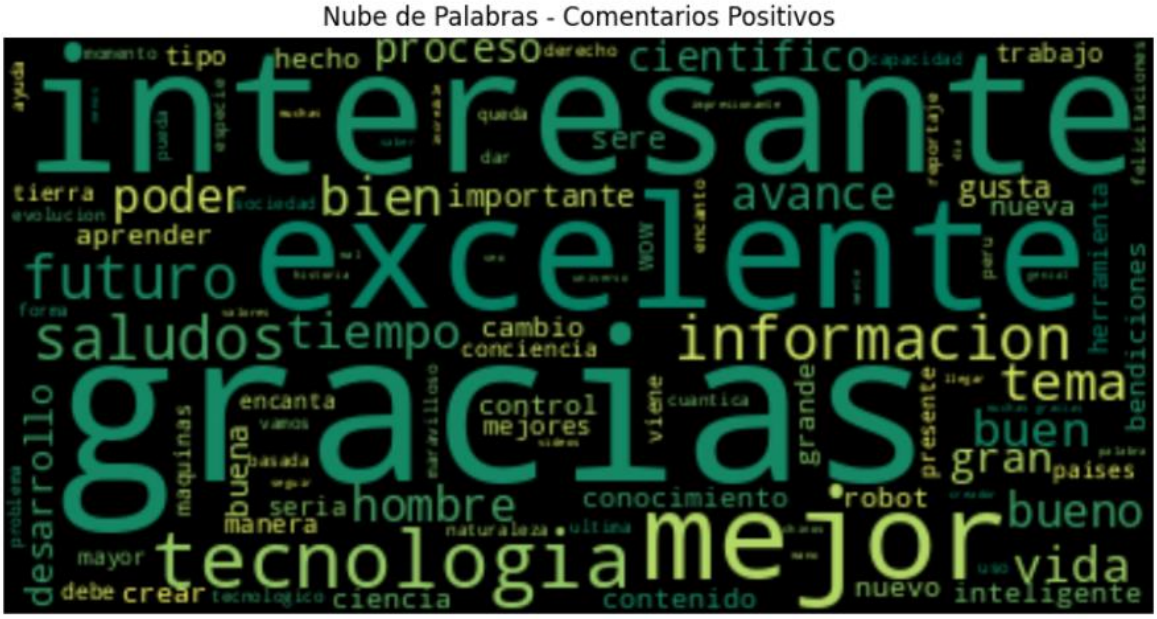

Ilustración 33. Nube de palabras sobre la clase positiva

Nube de palabras sobre comentarios negativos

Esta nube de palabras muestra y refleja las ideas negativas de los usuarios sobre la inteligencia artificial. A primera vista se siente una preocupación y escepticismo sobre el tema, la palabra más predominante es problema que muestra el temor de los usuarios

sobre el impacto que tiene esta herramienta, otras palabras como mal, miedo y control demuestran la percepción negativa, también se ven palabras como: futuro, dinero y empresas, que hablan de la intranquilidad de los usuarios sobre como esto genera cambios radicales en el empleo. Lo más inquietante de esta nube de palabras es que también se tiene una visión caótica, las palabras como guerra y control sugieren que los usuarios piensan que existirá o existe un posible descontrol ocasionado por la herramienta, en general esta nube de palabras refleja las quejas, inquietudes, temores y preocupaciones de los usuarios.

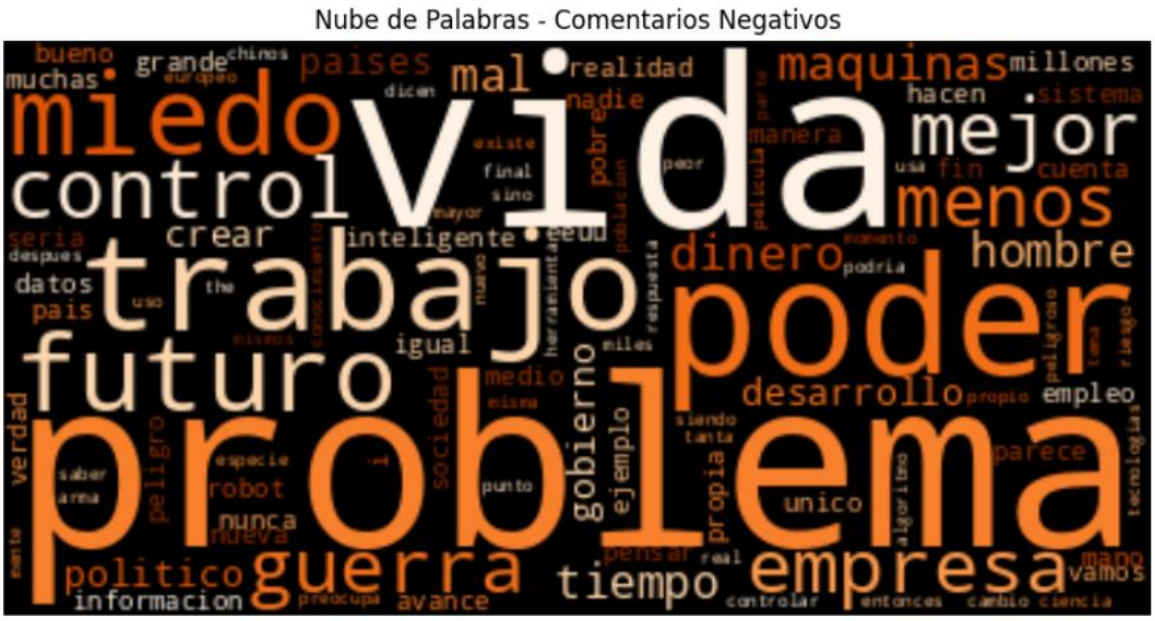

Ilustración 34. Nube de palabras sobre la clase negativa

Nube de palabras sobre los comentarios neutrales

Esta nube presenta palabras que reflejan observaciones y conceptos que no tienen una gran carga emocional, es decir, no se tiene una postura definida. Las palabras como empresa y sociedad tienen un enfoque de donde y quienes han estado usando e implementando la herramienta mundialmente, otras palabras como vida y poder reflejan como la inteligencia artificial tiene influencia en el aspecto laboral y económico. El hecho de tener palabras como importante, crear, evolución y desarrollo hablan del tema sin destacar preocupaciones o beneficios sino más bien hablando informativamente del

tema. Otras palabras como película y dios nos dan a entender que los usuarios hablan de casos hipotéticos o ideas que se han mostrado anteriormente en filmes.

Ilustración 35. Nube de palabras sobre la clase neutral

4.4. Modelado

4.4.1. Modelo Desbalanceado

Este modelo muestra como están distribuidos los datos en la vida real, debido a esto se tiene una distribución desigual de las clases de los datos, es decir, las etiquetas de positivo, negativo y neutro no tienen la misma cantidad representada, por lo que existe una clase mayoritaria. Debido a esto el modelo se desempeña mejor en esta clase predominante, mientras que en las otras clases minoritarias muestra un bajo rendimiento a la hora de predecirlos.

Con este modelo no se tienen resultados representativos que muestren una visión completa de las opiniones, esto porque las clases minoritarias son ignoradas por tener una clase mayoritaria, el modelo al tener pocos datos de las minorías se ajusta a la clase mayoritaria porque encuentra más patrones en esa clase y por lo tanto funciona mejor en esta, lo que resulta en un modelo que no predice correctamente.

4.4.2. Modelo balanceado con submuestreo

Al ser un modelo que toma la clase minoritaria y balancea aleatoriamente los datos de las etiquetas restantes para tener la misma cantidad en cada clase. Esto permite que el modelo pueda aprender de cada clase con datos representados en proporciones iguales, a diferencia del modelo desbalanceado este tiene una mejor capacidad de predicción, porque identifica y aprende de manera uniforme de los datos debido a que no existe una clase predominante, resultando en un modelo que identifica patrones de cada clase y por ende clasifica mejor.

4.4.3. Modelo balanceado con sobre muestreo - SMOTE

Este modelo se centra en sobre muestrear los comentarios, esto permite que pueda aprender patrones de una mayor cantidad de datos y a la vez balancearlo creando muestras sintéticas de las clases con menor cantidad de comentarios e igualando su cantidad a la clase mayoritaria, la razón por la que crea nuevos comentarios es para evitar duplicar los datos y generar un sobreajuste en el modelo.

4.4.4. Datos de prueba y entrenamiento

Antes de empezar con el modelado se deben preparar los datos de prueba y entrenamiento, primero convertimos el texto en vectores numéricos con la librería TF-IDF, se limita el análisis a las 3000 palabras más relevantes y genera una matriz “X” en la que cada fila representa un comentario y cada columna es el peso para una palabra especifica, es decir, en las columnas están las palabras seleccionadas por su peso de importancia en los comentarios, mientras que “y” es un vector numérico en la que cada elemento es un sentimiento de los comentarios en “X”.

La función fit_transform() analiza el texto para tener un vocabulario basado en las palabras más frecuentes o relevantes y poder calcular los valores del peso de cada palabra.

```
tfidf = TfidfVectorizer(max_features=3000)
X = tfidf.fit_transform(df['comentarios']).toarray()
y = df['sentimiento']
```

Ilustración 36. Código para convertir el texto en vectores numéricos

La decisión de elegir 3000 palabras relevantes permite que el modelo no tenga un exceso de ruido por palabras que son poco comunes o errores ortográficos de los comentarios y nos ayuda a asegurar que el análisis se centre en las palabras que son relevantes y aportan información al modelo.

Luego dividimos los datos para entrenamiento y prueba, se define que el 20% de los datos se asignan para el conjunto de prueba por lo que el 80% restante sirve para el conjunto de entrenamiento. La semilla de números aleatorios se usa para asegurar que se obtenga la misma división de datos cada vez que se ejecute el código, mientras que la función stratify() asegura que el conjunto de entrenamiento y de prueba mantengan la misma proporción de las clases de los datos originales.

```
#Dividir los datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

Ilustración 37. Código para dividir los datos de entrenamiento y prueba

4.4.5. Modelo Decision Tree

Desbalanceado

El modelo de árbol de decisión que fue implementado con la librería scikit learn obtuvo una precisión general del 58%, este bajo rendimiento se debe a que se trabajan con datos desbalanceados por lo que el modelo favorece a la clase mayoritaria y no clasifica correctamente las clases minoritarias, este tipo de modelos con datos desbalanceados afectan la calidad de las predicciones.

La matriz de confusión refleja esa clara desigualdad de predicción entre clases, la clase NEG como era de esperarse es la que tiene mayor precisión en predicciones con un valor de 228 predicciones correctas mientras que la clase POS tiene le peor rendimiento con un valor de 29 predicciones verdaderas.

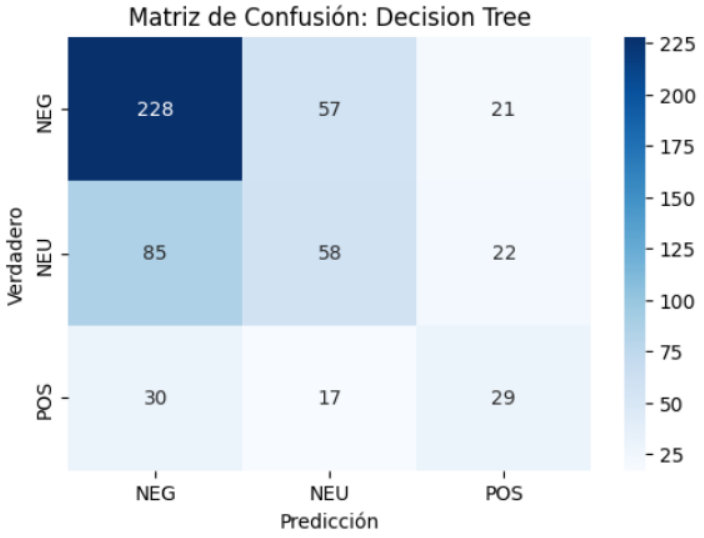

Ilustración 38. Matriz de confusión del modelo Decision Tree desbalanceado

Balanceado con submuestreo

Este modelo de árbol de decisión implementado con scikit learn y balanceado con submuestreo muestra un rendimiento más moderado de predicciones correctas en comparación al modelo desbalanceado, su precisión general es de 55% lo cual es bajo en general, la matriz de confusión refleja que el modelo tiene confusión a la hora de clasificar entre las clases pero que existe un mejor balance sin favorecer significativamente a una clase sobre otra, clasifico correctamente 41 ejemplos en la clase

NEG, 38 en NEU y 46 en la clase POS.

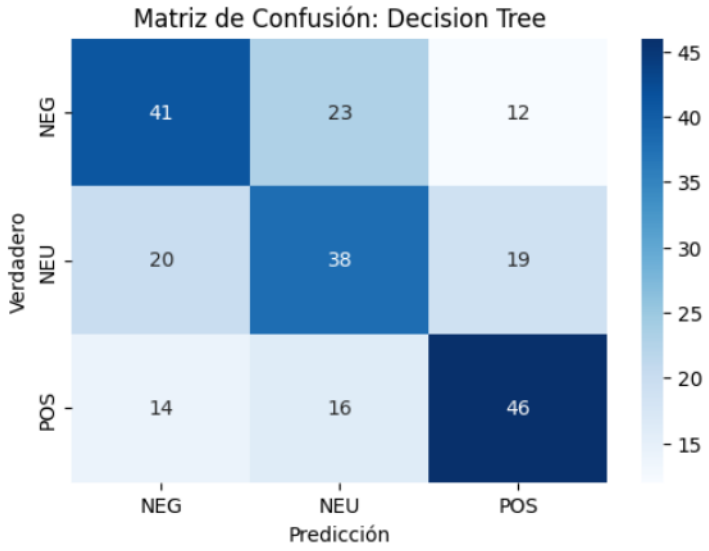

Ilustración 39. Matriz de confusión del modelo Decision Tree balanceado con submuestreo

Balanceado con sobre muestreo

El modelo de árbol de decisión implementado con scikit learn y balanceado con sobre muestreo con la herramienta SMOTE de imbalanced learn dio como resultado una precisión general del 71%, este valor es más alto en comparación al modelo desbalanceado y con submuestreo, esto se debe a que existen más datos de los cuales el modelo puede aprender y por ende este modelo tiende a predecir de mejor manera. La matriz de confusión muestra que se predijeron correctamente 203 comentarios negativos, 203 neutros y 248 positivos, predice ligeramente mejor a la clase positiva en comparación a las demás. En este modelo se tiene confusión a la hora de predecir los comentarios negativos y neutros, en la clase negativa esto se refleja con valores de 78 negativos predichos como neutros y 25 como positivos y en la clase neutra con valores de 80 neutros clasificados como negativos y 22 como positivos.

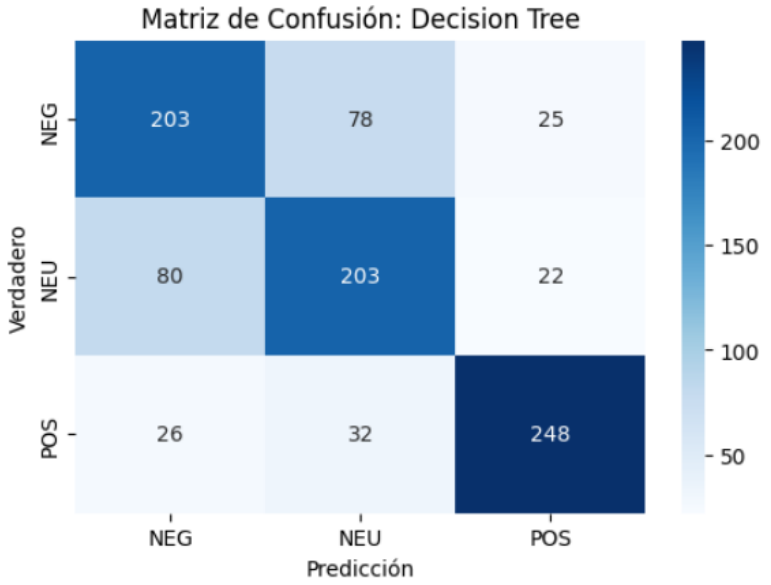

Ilustración 40. Matriz de confusión del modelo Decision Tree balanceado con sobre muestreo

4.4.6. Modelo Random Forest

Desbalanceado

Este modelo de random forest implementado con la librería scikit learn dio como resultado una precisión general de 62%, en la matriz de confusión se presenta la clara desigualdad de clasificación en las predicciones, esto se debe a que los datos utilizados están desbalanceados, es decir que se tiene una clase mayoritaria y minoritaria, la clase NEG (mayoritaria) tiene la mayor precisión en predicciones con un valor de 280 predicciones correctas y la clase minoritaria POS tiene el peor rendimiento con un valor de 23 predicciones verdaderas y la clase NEU con 38 predicciones.

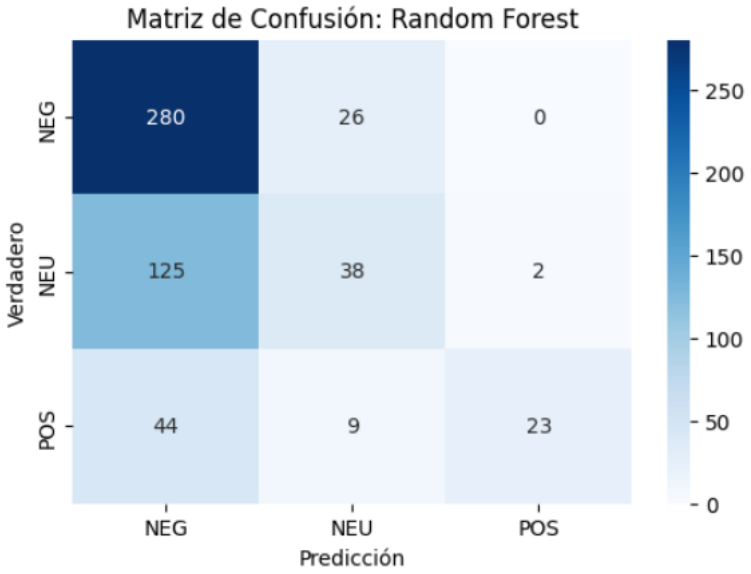

Ilustración 41. Matriz de confusión del modelo Random Forest desbalanceado

Balanceado con submuestreo

El modelo de random forest implementado con scikit learn y balanceado con submuestreo muestra una clara igualdad entre la clasificación de clases a la hora de predecir, se obtuvo una precisión general del 63%, la matriz de confusión refleja un rendimiento solido a la hora de clasificar los comentarios en las diferentes clases, el modelo predijo correctamente 54 comentarios en la clase NEG, 49 e la clase NEU y 42 en la POS, en este modelo se presenta una confusión a la hora de clasificar los positivos, dentro de esta clase se predijo 25 como neutros y 9 como negativos.

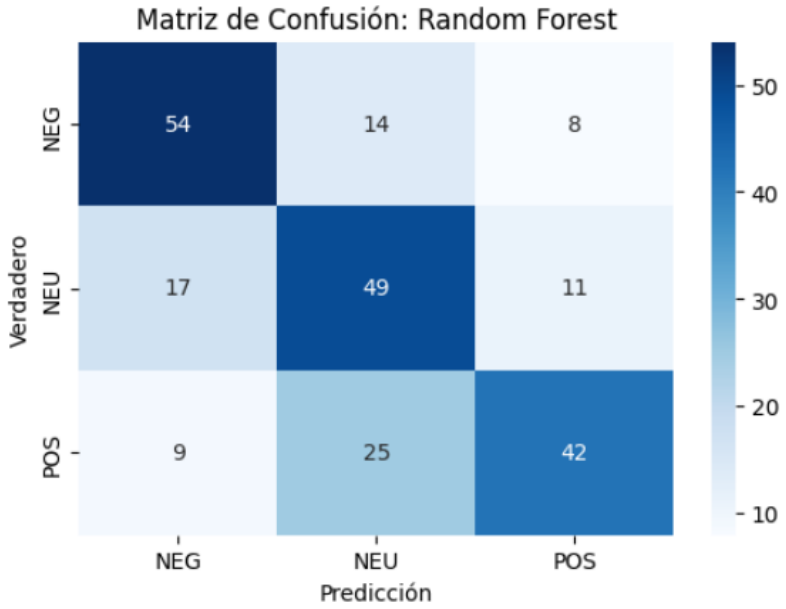

Ilustración 42. Matriz de confusión del modelo Random Forest balanceado con submuestreo

Balanceado con sobre muestreo

El modelo de random forest implementado con scikit learn y balanceado con sobre muestreo con la herramienta SMOTE de imbalanced learn dio como resultado una precisión general del 82%, este valor es mayor en comparación al modelo desbalanceado y con submuestreo debido a que hay más datos de los cuales el modelo puede aprender y como resultado el modelo predice mejor las clases.

La matriz de confusión presenta los siguientes resultados: se predijo correctamente 252 comentarios negativos, 226 neutros y 270 positivos, las clases positiva y negativa tiene una predicción similar mientras que la neutra es un poco menos exacta en comparación. En este modelo se tiene confusión al predecir la clase neutra, se predijo 72 neutros como negativos y 7 como positivos.

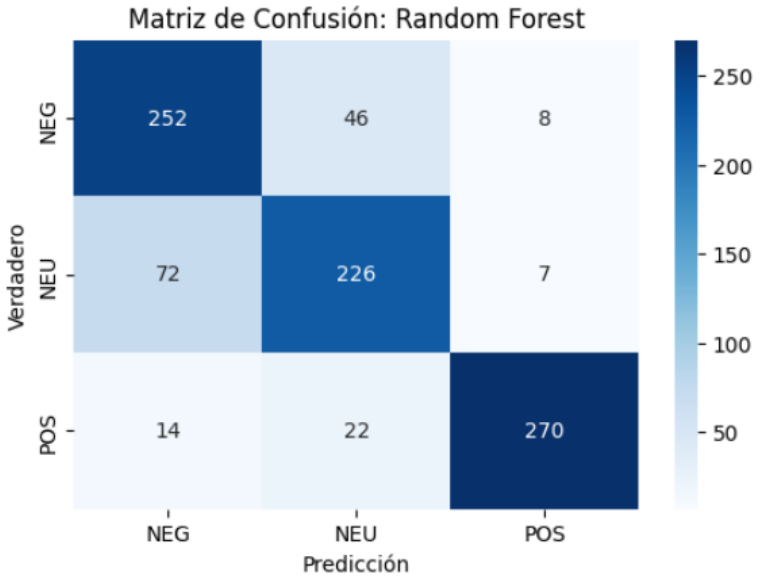

Ilustración 43. Matriz de confusión del modelo Random Forest balanceado con sobre muestreo

4.4.7. Modelo Support Vector Machine

Desbalanceado

Este modelo SVM creado con la librería scikit learn dio como un resultado una precisión general del 69%, muestra una desigualdad entre clases favoreciendo a la mayoritaria por el hecho de usar datos desbalanceados, la matriz de confusión muestra claramente la cantidad de precisión que muestra la clase mayoritaria (NEG) en comparación a las otras dos clases, la clase NEG tiene un valor de 278 predicciones verdaderas, la clase NEU tiene un valor de 67 predicciones y la clase POS tiene 31 predicciones correctas.

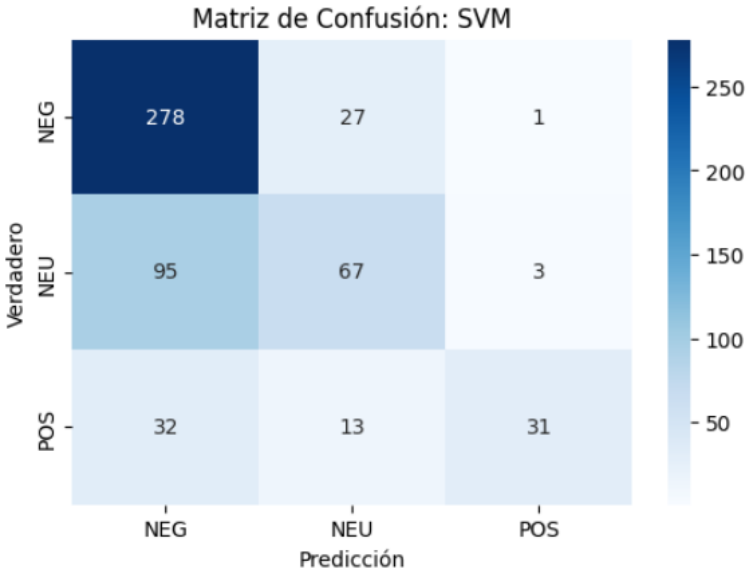

Ilustración 44. Matriz de confusión del modelo SVM desbalanceado

Balanceado con submuestreo

El modelo de Support Vector Machine implementado con la librera scikit learn y balanceado con submuestreo muestra un balance entre la clasificación de clases, tiene una precisión general de 67%, la matriz de confusión muestra que se predijo correctamente 53 comentarios como negativos, 52 como neutros y 46 positivos.

En este modelo se presenta una confusión al predecir los positivos, dentro de esta clase se predijeron 11 positivos como negativos y 19 como neutros, aunque de manera general los valores predichos incorrectamente varían por muy poco entre las clases.

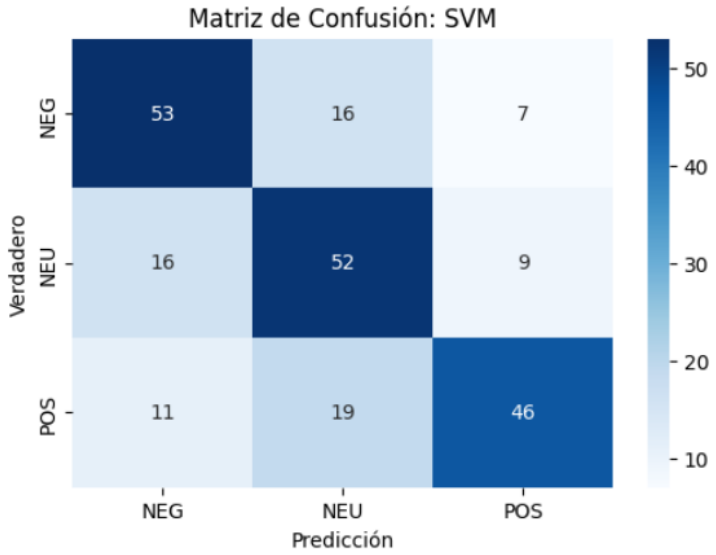

Ilustración 45. Matriz de confusión del modelo SVM balanceado con submuestreo

Balanceado con sobre muestreo

El modelo de SVM implementado con scikit learn y balanceado con sobre muestreo con la herramienta SMOTE de imbalanced learn dio como resultado una precisión general del 79%, la matriz de confusión muestra los siguientes resultados: se predijo correctamente 220 comentarios negativos, 239 neutros y 264 positivos, las clases positiva y neutra tiene una predicción similar mientras que la negativa es un poco menos exacta en comparación. En este modelo se tiene confusión al predecir la clase negativa, se predijo 65 negativos como neutros y 21 como positivos.

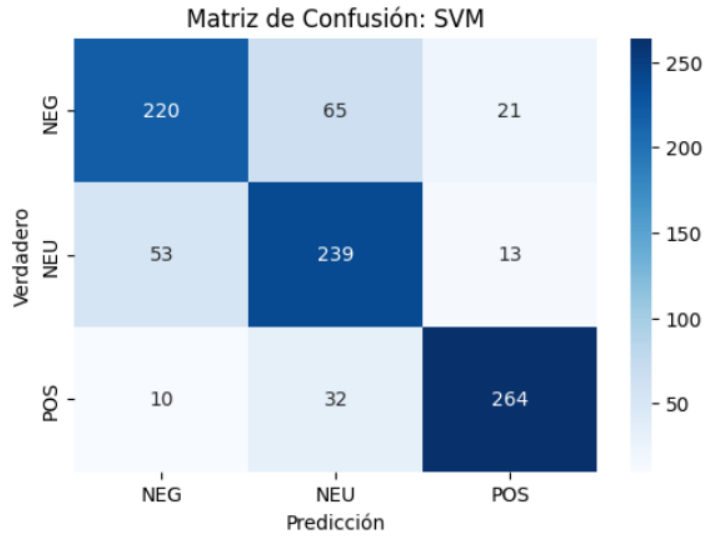

Ilustración 46. Matriz de confusión del modelo SVM balanceado con sobre muestreo

4.4.8. Modelo de Regresión Logística

Desbalanceado

Para este modelo de regresión logística creado con la librería scikit learn se obtuvo una precisión general del 68% para los datos desbalanceados, la matriz de confusión muestra claramente una diferencia de precisión de las predicciones entre clases, la clase NEG se predijo correctamente 283 veces, la clase NEU predijo 62 y la clase POS 29, esto demuestra que los modelos con datos desbalanceados presentan desigualdad en las predicciones favoreciendo indiscutiblemente a la clase mayoritaria.

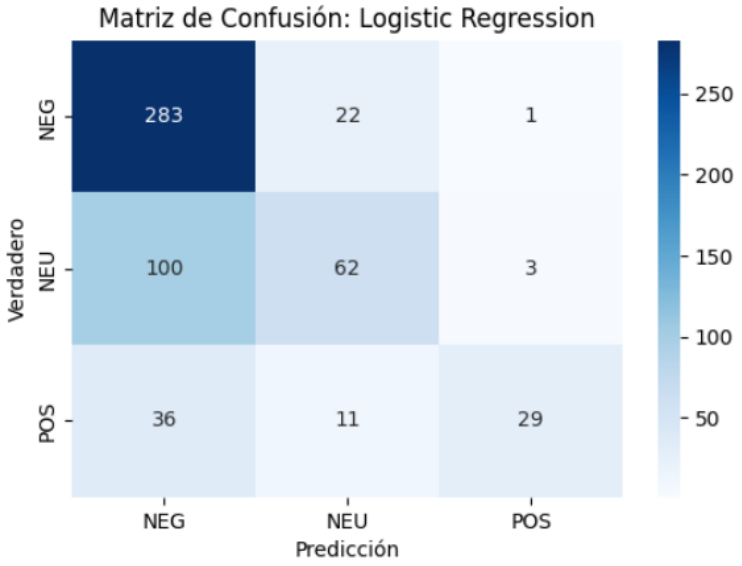

Ilustración 47. Matriz de confusión del modelo Logistic Regression desbalanceado

Balanceado con submuestreo

El modelo de regresión logística implementado con scikit learn y balanceado con submuestreo obtuvo una precisión general de 63%, esta muestra un rendimiento de clasificación más equitativo entre clases que los modelos desbalanceados, la matriz de confusión presenta que la clase negativa se predijo 50 veces correctamente, 49 en la clase neutra y 46 en la positiva. Los errores que más ocurren son al predecir los positivos, dentro de esta clase se predijeron 9 positivos como negativos y 21 como neutros, aunque de manera general los valores predichos incorrectamente entre cada clase varían por muy poco.

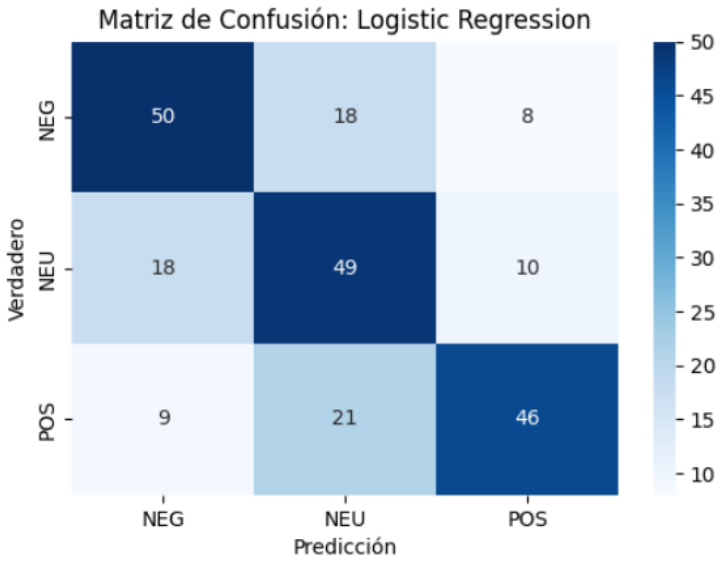

Ilustración 48. Matriz de confusión del modelo Logistic Regression balanceado con submuestreo

Balanceado con sobre muestreo

El modelo de regresión logística implementado con scikit learn y balanceado con sobre muestreo con la herramienta SMOTE de imbalanced learn dio como resultado una precisión general del 76%, la matriz de confusión refleja una predicción igualitaria para las tres clases, se predijo correctamente 214 comentarios negativos, 229 neutros y 258 positivos, la clase positiva es un poco más exacta en comparación a las clases negativa y neutra que tiene una predicción similar. En este modelo se tiene confusión al predecir la clase de los negativos, se predijo 63 negativos como neutros y 29 como positivos.

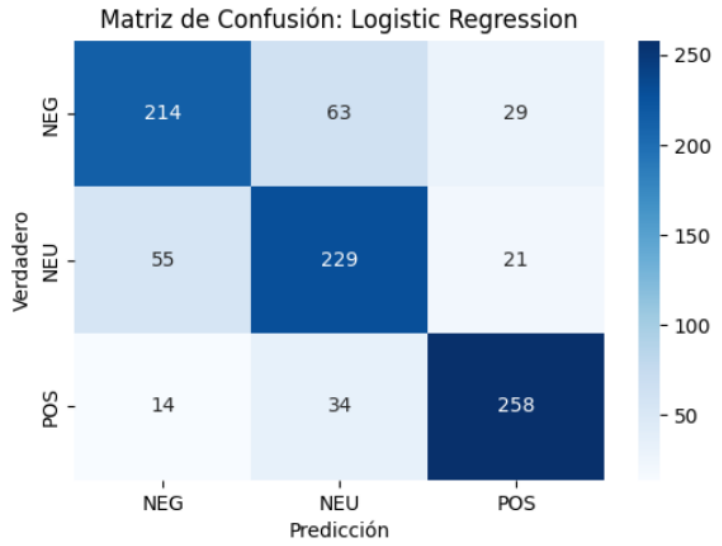

Ilustración 49. Matriz de confusión del modelo Logistic Regression balanceado con sobre muestreo

4.5. Evaluación de los modelos

4.5.1. Tabla comparativa del Support para los modelos

Métrica	Clase	Desbalanceado	Submuestreo	SMOTE
Support	Negativo	306	76	306
	Neutral	165	76	306
	Positivo	76	76	306

Tabla 1. Tabla comparativa de la métrica Support para los modelos

4.5.2. Tabla comparativa de los modelos

Modelo	Métrica	Desbalanceado	Submuestreo	SMOTE
Decision Tree	Precisión	0.56	0.55	0.72
	Recall	0.58	0.55	0.71
	F1-Score	0.57	0.55	0.71
Random Forest	Precisión	0.63	0.64	0.82
	Recall	0.62	0.63	0.82
	F1-Score	0.57	0.63	0.82
SVM	Precisión	0.70	0.67	0.79
	Recall	0.69	0.66	0.79
	F1-Score	0.66	0.66	0.79
Logistic Regression	Precisión	0.70	0.64	0.77
	Recall	0.68	0.63	0.76
	F1-Score	0.65	0.63	0.76

Tabla 2. Tabla comparativa de todos los modelos

5. Conclusiones y Recomendaciones

5.1. Conclusiones

- Aunque el balanceo de datos con submuestreo mejoró la clasificación equitativa de las clases, disminuyó la precisión en general por el hecho de que al tener menor cantidad de datos el modelo tiene menos casos de los cuales aprender, como resultado no tiene una gran capacidad predictiva porque no puede generalizar. Por otro lado, el modelo con sobre muestreo dejo claro que si un modelo tiene más datos de los cuales aprender las predicciones serán mejores.
- Los modelos desbalanceados favorecieron completamente a la clase con la mayor cantidad de datos mientras que las clases balanceadas con submuestreo y sobre muestreo mejoraron significativamente la clasificación de los sentimientos que en un inicio eran clases minoritarias, sin mencionar el aumento de precisión general en ambos casos.
- Traducir los comentarios a un solo idioma mejoro la calidad de las predicciones, al tener un solo idioma se puede evitar el ruido generado por otros, pero como desventaja se tiene la incertidumbre de si la traducción va o no a capturar el significado completo de lo que el usuario quiso decir.
- La librería de pysentimiento fue eficiente al etiquetar o clasificar los sentimientos de los comentarios que habían sido traducidos, esto gracias a que es capaz de considerar el contexto e interpretar textos más casuales además de su soporte multilingüe que incluye el español.
- A pesar de que el modelo desbalanceado tuvo un peor rendimiento y favoreció únicamente a la clase mayoritaria es el reflejo de los resultados que se obtendrían en el mundo real mientras que los datos balanceados son una representación falsa que da la misma importancia a todas las clases, es decir, no es realista.
- Entre los modelos utilizados gracias a las tablas comparativas se puede saber que sobre muestreo (SMOTE) es la mejor herramienta para balancear los datos y

permitir que el modelo pueda detectar las clases minoritarias con mayor facilidad.

- Los modelos de SVM y regresión logística son los que tuvieron un mejor rendimiento que fue casi similar con los datos desbalanceados, con el submuestreo el modelo que mejor rendimiento obtuvo fue el de SVM, el modelo que mejor rendimiento obtuvo con el sobre muestreo fue el modelo de Random Forest y el modelo que tuvo más limitaciones en su rendimiento en general con los tres tipos de datos fue el modelo de Árbol de decisión.

5.2. Recomendaciones

- Es aconsejable representar mejor las clases minoritarias, para futuros análisis similares se podría priorizar la misma u otra técnica que aumente los datos para que las clases con menos datos no sean ignoradas, pero a la vez evitar técnicas de aumento de datos con duplicación para no llegar a obtener un posible sobreajuste que resulte en un modelo que no puede generalizar.
- Es recomendable tener más datos en el dataset inicial porque puede ayudar a mejorar el rendimiento de los modelos y se podrían aplicar las técnicas de submuestreo para evitar los efectos que se presentaron por tener pocos datos de los cuales el modelo podía aprender.
- Para mejorar el rendimiento se puede realizar una comparación del mismo trabajo, pero con otros idiomas y tomando únicamente fuentes de datos de un solo idioma para evaluar si realmente la traducción afecta la precisión del análisis. Esto porque por lo general se menciona que las traducciones automáticas hacen que el comentario pierda el sentido original por factores como los modismos, el contexto original y demás.
- Se sugiere utilizar herramientas que nos ayudan a encontrar la mejor combinación de los parámetros que tiene los modelos para realizar una optimización y obtener mejores resultados, se pueden probar una y otra vez hasta que se obtenga el

mejor rendimiento posible.

- Se sugiere explorar otros tipos de modelos que puedan manejar mejor los datos complejos, con el fin de identificar cual puede ser el modelo más adecuado para este tipo de análisis.
- Para entender más a fondo las opiniones de la inteligencia artificial es enfocar este análisis a subtemas como la ética y el impacto laboral que tiene, y se podrían incluir más herramientas analíticas que nos ayuden a profundizar en los datos obtenidos para encontrar insights que en un futuro puedan ser de guía para minimizar su impacto
- Sugerir la implementación de este tipo de trabajo dentro de las herramientas de análisis de sistemas de gestión dentro de las organizaciones para fomentar un uso orientado a cualquier tipo de tema con el fin de entender que es lo que los usuarios finales de ese producto o servicio opinan de la organización.

6. Bibliografía

Agarwal, B., Nayak, R., Mittal, N., & Patnaik, S. (2020). Deep learning-based approaches for sentiment analysis. *Deep learning-based approaches for sentiment analysis.* Springer, Kallang, Singapore. Obtenido de https://books.google.com.ar/books?id=tSTMDwAAQBAJ&pg=PR6&dq=sentiment+analysis&hl=es&sa=X&ved=2ahUKEwivhdf7wpLuAhWJHLkGHTpqDZgQ6AEwAXoECAQQAg#v=onepage&q=sentiment%20analysis&f=false

analyticswithadam. (27 de October de 2023). *YouTube Comments Advanced*. Obtenido de GitHub: https://github.com/analyticswithadam/Python/blob/main/YouTube_Comments_Advanced.ipynb

Cornejo Urbina, F. M. (2022). El poder de la comunicación: medios, política y ciudadanos. *Comuni@cción*, 74-85.

Google. (21 de Julio de 2022). *YouTube Data API v3*. Obtenido de Google Cloud: https://console.cloud.google.com/apis/library/youtube.googleapis.com?hl=es

Hiremath S, O. (24 de October de 2024). *Edureka*. Obtenido de Edureka: https://www.edureka.co/blog/web-scraping-with-python/

IBM. (17 de Agosto de 2021). *CRISP-DM process diagram.* Obtenido de IBM: https://www.ibm.com/docs/es/spss-modeler/saas?topic=dm-crisp-help-overview

imbalanced-learn, d. (24 de Dic de 2024). *SMOTE*. Obtenido de imbalanced learn: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

Isasi, J. (2021). Análisis de sentimientos en R con'syuzhet'. *The Programming Historian en español*, N/A.

Matplotlib, D. (s.f. de s.f. de 2024). *Matplotlib: Visualization with Python*. Obtenido de Matploplib: https://matplotlib.org/

Medhat, W.; Hassan, A.; Korashy, H. (April de 2014). Sentiment analysis algorithms and

applications. *Ain Shams engineering journal*, 1093-1113.

Müller, A. (s.f. de s.f. de 2020). *WordCloud for Python documentation*. Obtenido de WordCloud: https://amueller.github.io/word_cloud/

Narváez U., D. (2015). ¿Qué son los sentimientos? *¿Qué son los sentimientos?* Universidad Los Leones(ULL), Santiago, Chile.

NLTK. (14 de Noviembre de 2024). *Kit de herramientas de lenguaje natural*. Obtenido de NLTK : https://www.nltk.org/

pandas development team. (20 de Sep de 2024). *pandas*. Obtenido de pandas: https://pandas.pydata.org/

Pérez, A. (27 de October de 2021). *pysentimiento: A Python toolkit for Sentiment Analysis and Social NLP tasks*. Obtenido de GitHub: https://github.com/pysentimiento/pysentimiento

PyPI. (01 de Enero de 2025). *googletrans*. Obtenido de Python Package Index: https://pypi.org/project/googletrans/

Python. (14 de Noviembre de 2024). *Unicode HOWTO*. Obtenido de Python : https://docs.python.org/3/howto/unicode.html

Rosero Correa, L. E. (2021). Desarrollo de un framework que identifica, describe y organiza recursos educativos disponibles en plataformas web de universidades mediante minería de datos. *Desarrollo de un framework que identifica, describe y organiza recursos educativos disponibles en plataformas web de universidades mediante minería de datos.* Escuela Politécnica Nacional, Quito.

Scikit Learn. (14 de Noviembre de 2024). *scikit-learn*. Obtenido de Scikit Learn: https://scikit-learn.org/stable/

Scikit-learn, d. (24 de Dic de 2024). *TfidfVectorizer*. Obtenido de Scikit-learn: https://scikit-learn.org/1.5/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Sharma, H. D., & Goyal, P. (2023). An Analysis of Sentiment: Methods, Applications, and

Challenges. *Engineering Proceedings*, 59(1), 68.

Waskom, M. (s.f. de s.f. de 2024). *Seaborn: statistical data visualization*. Obtenido de Seaborn: https://seaborn.pydata.org/tutorial/introduction.html