```
import pandas as pd
import numpy as np
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Instalaciones

[ ] ↪ *13 celdas ocultas*

Diccionario Emocional

[ ] ↪ *20 celdas ocultas*

PreProcesamiento de Dataset de Comentarios

[ ] ↪ *9 celdas ocultas*

Stop Words

```
conjunto_de_dato2=conjunto_de_datos[conjunto_de_datos['body'] == '[deleted]'].index.to_list()
# Filtramos los comentarios que no contienen la frase no deseada
conjunto_de_datos = conjunto_de_datos.drop(conjunto_de_dato2)
conjunto_de_datos = conjunto_de_datos[~conjunto_de_datos['body'].str.contains("Your post has been removed because it violates
```

```
import nltk
from nltk.corpus import stopwords
import pandas as pd

# Descargar la lista de palabras de parada (stop words) si aún no lo has hecho
nltk.download('stopwords')

# Crear una lista de stop words en el idioma que desees, por ejemplo, en español
stop_words = set(stopwords.words('english'))

# Suponiendo que tienes una columna 'content_clean' en tu DataFrame 'data'
conjunto_de_datos['Cleaned Body'] = conjunto_de_datos['Cleaned Body'].apply(lambda text: ' '.join([word for word in text.spli

# Esto eliminará las stop words de cada fila en la columna 'content_clean'

# Imprime el DataFrame resultante
conjunto_de_datos.head()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

	body	id	score	created_utc	created_datetime	Cleaned Body
0	I'd like to see this sub be more active, too. ...	cqowxhs	1	1430023102	2015-04-26 04:38:22	like see sub active across reddit lots subredd...
1	I've found people are more receptive when you ...	cvzg3v2	1	1444835103	2015-10-14 15:05:03	found people receptive take responsibility sho...
2	Thank you so much. I have been trying to use m...	cw65vo8	1	1445326215	2015-10-20 07:30:15	thank much trying use phone set alerts reminde...
4	Sooooo, not sure why you were told it was					sooooo sure told hours release get hours

```
conjunto_de_datos_bckp = conjunto_de_datos.copy()
len(conjunto_de_datos_bckp)
```

```
199785
```

```
conjunto_de_datos = conjunto_de_datos_bckp.copy()  # Obtener una muestra aleatoria de 50000 filas
len(conjunto_de_datos)
```

```
199785
```

```
conjunto_de_datos = conjunto_de_datos_bckp.sample(n=50000)  # Obtener una muestra aleatoria de 50000 filas
len(conjunto_de_datos)
```

```
50000
```

BERT

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
import dask
import dask.dataframe as dd
import pandas as pd
from nltk import word_tokenize
from transformers import BertTokenizer, BertModel
import torch
import numpy as np
import spacy

# Configuración para usar GPU
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Cargar el tokenizador y modelo BERT preentrenado en GPU
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')
model.to(device)

# Cargar el modelo de lematización de SpaCy para inglés
nlp = spacy.load('en_core_web_sm')

# Convierte tu DataFrame de pandas a un DataFrame de Dask
dask_dataframe = dd.from_pandas(conjunto_de_datos, npartitions=8)

# Define una función para tokenizar el texto
def tokenize_text(text):
    return word_tokenize(text)

# Define una función para lematizar el texto utilizando SpaCy
def lemmatize_text(text):
    doc = nlp(text)
    lemmatized_tokens = [token.lemma_ for token in doc]
    return lemmatized_tokens

# Define una función para obtener embeddings de palabras en GPU
def get_word_embeddings(tokens):
    if not tokens:  # Manejar casos donde no hay tokens
        return np.zeros((1, model.config.hidden_size))
    inputs = tokenizer(tokens, return_tensors='pt', padding=True, truncation=True, max_length=512)
    inputs.to(device)
    with torch.no_grad():
        outputs = model(**inputs)
    # Solo tomamos la representación de la última capa oculta (puedes experimentar con otras capas)
    embeddings = outputs.last_hidden_state.mean(dim=1).cpu().numpy()  # Promedio sobre las dimensiones de la secuencia
    return embeddings

torch.cuda.empty_cache()

# Aplica la tokenización en paralelo a la columna 'Cleaned Body'
dask_dataframe['Tokenized Body'] = dask_dataframe['Cleaned Body'].map(tokenize_text, meta=('Tokenized Body', 'object'))

# Aplica la lematización en paralelo a la columna 'Cleaned Body'
dask_dataframe['Lemmatized Body'] = dask_dataframe['Cleaned Body'].map(lemmatize_text, meta=('Lemmatized Body', 'object'))

# Aplica la función para obtener embeddings en paralelo a la columna 'Tokenized Body'
dask_dataframe['Word Embeddings'] = dask_dataframe['Tokenized Body'].map(get_word_embeddings, meta=('Word Embeddings', 'obj

# Aplica la función para obtener embeddings en paralelo a la columna 'Lemmatized Body'
dask_dataframe['Lemmatized Word Embeddings'] = dask_dataframe['Lemmatized Body'].map(get_word_embeddings, meta=('Lemmatized

# Calcula el resultado del DataFrame
conjunto_de_datos = dask_dataframe.compute(scheduler='threads')
display(conjunto_de_datos)
```

	body	id	score	created_utc	created_datetime	Cleaned Body	Tokenized Body	Lemmatized Body	Word Embeddings	Lemmatize Wo Embeddin
2	Thank you so much. I have been trying to use m...	cw65vo8	1	1445326215	2015-10-20 07:30:15	thank much trying use phone set alerts reminde...	[thank, much, trying, use, phone, set, alerts,...	[thank, much, try, use, phone, set, alert, rem...	[[0.10868178, 0.42426407, 0.24624643, 0.089167...	[[0.204236 0.256009 0.215510 0.01724
5	My doctor is reluctant to give me	d38f9y3	1	1463458428	2016-05-17 04:13:48	doctor reluctant give fast acting	[doctor, reluctant, give, fast,	[doctor, reluctant, give, fast,	[[0.029582903, 0.27469778, 0.17197813	[[0.029582 0.274697 0.171978

Emocion Diccionario

```
import dask.dataframe as dd
import pandas as pd
from nltk import word_tokenize

# Convierte tu DataFrame de pandas a un DataFrame de Dask
dask_dataframe = dd.from_pandas(conjunto_de_datos, npartitions=8)  # Puedes ajustar el número de particiones según tus nece

# Define una función para tokenizar el texto
def tokenize_text(text):
    return word_tokenize(text)

# Aplica la tokenización en paralelo a la columna 'Cleaned Body'
#dask_dataframe['Tokenized Body'] = dask_dataframe['Cleaned Body'].map(tokenize_text, meta=('Cleaned Body', 'object'))

# Define una función para etiquetar emociones
# Define una función para etiquetar emociones
def etiquetar_emocion(comentario):
    emocion_predominante = None
    probabilidad_maxima = 0.0
    if isinstance(comentario, list):
        comentario = " ".join(comentario)
    for emocion, palabras_probabilidades in nuevo_diccionario_emocional.items():
        probabilidad_total = 1.0

        palabras_comentario = comentario.split()  # Tokenizar el comentario si no está tokenizado

        for palabra in palabras_comentario:
            if palabra in palabras_probabilidades:
                probabilidad_total *= palabras_probabilidades[palabra]

        if probabilidad_total > probabilidad_maxima:
            probabilidad_maxima = probabilidad_total
            emocion_predominante = emocion

    return emocion_predominante


# Aplica la función de etiquetado de emoción en paralelo a la columna 'Tokenized Body'
dask_dataframe['emocion_predominante'] = dask_dataframe['Tokenized Body'].map(etiquetar_emocion, meta=('Tokenized Body', 'o

# Calcula el resultado del DataFrame
conjunto_de_datos = dask_dataframe.compute(scheduler='threads')  # Puedes ajustar el planificador según tus necesidades (po
conjunto_de_datos.head()
# Ahora deberías obtener los resultados correctamente en 'conjunto_de_datos'.
```

	body	id	score	created_utc	created_datetime	Cleaned Body	Tokeniz Bc
2	Thank you so much. I have been trying to use m...	cw65vo8	1	1445326215	2015-10-20 07:30:15	thank much trying use phone set alerts reminde...	[tha mu trying, u phone, s alerts
5	My doctor is reluctant to give me a fast actin...	d38f9y3	1	1463458428	2016-05-17 04:13:48	doctor reluctant give fast acting dosage later...	[doc reluct give, fa acti dosag
	That's not					good would	[go

```
len(conjunto_de_datos.loc[conjunto_de_datos['emocion_predominante'] == 'empty'])
```

```
48908
```

```
print(len(conjunto_de_datos['Word Embeddings']))
print(len(conjunto_de_datos))
```

```
50000
50000
```

```
from gensim.models import KeyedVectors
import gensim.downloader as api

# Descargar el modelo Word2Vec preentrenado de Google News
word_embeddings_model = api.load('word2vec-google-news-300')
```

```
from sklearn.cluster import KMeans
import numpy as np
import gensim.downloader as api

# Número de clusters (ajusta según tus necesidades)
num_clusters = 6

# Descargar el modelo Word2Vec preentrenado de Google News
word_embeddings_model = api.load('word2vec-google-news-300')

#embedding = conjunto_de_datos['Lemmatized Body']

X = np.array([np.mean([word_embeddings_model[word.lower()] for word in embedding if word.lower() in word_embeddings_model],
```

```
# Ajustar el modelo de KMeans
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
kmeans.fit(X)

# Asignar etiquetas de cluster a tus datos
conjunto_de_datos['Cluster'] = kmeans.labels_
# Crear una nueva columna llamada 'Vector' en conjunto_de_datos
conjunto_de_datos['Vector'] = [np.mean([word_embeddings_model[word.lower()] for word in embedding if word.lower() in word_e

# Mostrar los resultados
display(conjunto_de_datos)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: Futur
  warnings.warn(
```

	body	id	score	created_utc	created_datetime	Cleaned Body	Tok
2	Thank you so much. I have been trying to	cw65vo8	1	1445326215	2015-10-20 07:30:15	thank much trying use phone set alerts	try ph
5	to give me	d38f9v3	1	1463458428	2016-05-17 04:13:48	give last	g

Balance

```
from sklearn.utils import resample

# Supongamos que ya tienes un DataFrame llamado conjunto_de_datos con una columna llamada 'Emocion'

# Dividir el DataFrame en subconjuntos para cada clase
empty_data = conjunto_de_datos[conjunto_de_datos['emocion_predominante'] == 'empty']
otras_emociones_data = conjunto_de_datos[conjunto_de_datos['emocion_predominante'] != 'empty']

# Sobreponderar la clase minoritaria (otras emociones) para que tenga el mismo número de muestras que la clase mayoritaria
otras_emociones_sobreponderado = resample(otras_emociones_data, replace=True, n_samples=len(empty_data), random_state=42)

# Combinar los subconjuntos para crear un nuevo DataFrame equilibrado
conjunto_de_datos_equilibrado = pd.concat([empty_data, otras_emociones_sobreponderado])

# Mostrar el DataFrame equilibrado
conjunto_de_datos_equilibrado.head()
```

	body	id	score	created_utc	created_datetime	Cleaned Body	Tokenized Body	Tokenized Body Bck	emocion_predominante
0	I'd like to see this sub be more active, too. ...	cqowxhs	1	1430023102	2015-04-26 04:38:22	like see sub active across reddit lots subredd...	[like, see, sub, active, across, reddit, lots,...	[like, see, sub, active, across, reddit, lots,...	empty
	I've found people are					found people			

Conteo de Palabras

Ansiedad

```
import pandas as pd

# Supongamos que ya tienes un DataFrame llamado conjunto_de_datos con una columna llamada 'Tokenized Body'

# Crear una función para contar las veces que aparece 'anxiety' o 'stress' en un texto
def contar_anxiety_stress(texto):
    return texto.count('anxiety')

# Aplicar la función a la columna 'Tokenized Body' y crear una nueva columna 'anxiety/stress'
conjunto_de_datos_equilibrado['anxiety'] = conjunto_de_datos_equilibrado['Tokenized Body'].apply(contar_anxiety_stress)

# Sumar el total de 'anxiety/stress'
total_anxiety_stress = conjunto_de_datos_equilibrado['anxiety'].sum()

# Imprimir el resultado
print(f"Total de veces que aparece 'anxiety': {total_anxiety_stress}")
```

```
Total de veces que aparece 'anxiety': 50900
```

Depresion

```
import pandas as pd

# Supongamos que ya tienes un DataFrame llamado conjunto_de_datos con una columna llamada 'Tokenized Body'

# Crear una función para contar las veces que aparece 'anxiety' o 'stress' en un texto
def contar_anxiety_stress(texto):
    return texto.count('depression')

# Aplicar la función a la columna 'Tokenized Body' y crear una nueva columna 'anxiety/stress'
conjunto_de_datos_equilibrado['depression'] = conjunto_de_datos_equilibrado['Tokenized Body'].apply(contar_anxiety_stress)

# Sumar el total de 'anxiety/stress'
total_anxiety_stress = conjunto_de_datos_equilibrado['depression'].sum()

# Imprimir el resultado
print(f"Total de veces que aparece 'depression': {total_anxiety_stress}")
```

```
Total de veces que aparece 'depression': 31171
```

Hioersensibilidad Sensorial

```
import pandas as pd

# Supongamos que ya tienes un DataFrame llamado conjunto_de_datos con una columna llamada 'Tokenized Body'

# Crear una función para contar las veces que aparece 'anxiety' o 'stress' en un texto
def contar_anxiety_stress(texto):
    return texto.count('hypersensitivity') + texto.count('sensory')

# Aplicar la función a la columna 'Tokenized Body' y crear una nueva columna 'anxiety/stress'
conjunto_de_datos_equilibrado['hypersensitivity'] = conjunto_de_datos_equilibrado['Tokenized Body'].apply(contar_anxiety_st

# Sumar el total de 'anxiety/stress'
total_anxiety_stress = conjunto_de_datos_equilibrado['hypersensitivity'].sum()

# Imprimir el resultado
print(f"Total de veces que aparece 'hypersensitivity': {total_anxiety_stress}")
```

```
Total de veces que aparece 'hypersensitivity': 4577
```

```
import pandas as pd

# Supongamos que ya tienes un DataFrame llamado conjunto_de_datos con una columna llamada 'Tokenized Body'

# Crear una nueva columna 'Coincidencias' que contiene True si ambas palabras están presentes, False de lo contrario
conjunto_de_datos_equilibrado['textures_food'] = conjunto_de_datos_equilibrado['Tokenized Body'].apply(lambda x: 'textures'

# Filtrar el DataFrame para mostrar solo las filas donde hay coincidencias
coincidencias_df = conjunto_de_datos_equilibrado['textures_food']

# Imprimir el resultado
print(f"Total de veces que aparece 'textures' y 'food': {total_anxiety_stress}")
```

```
Total de veces que aparece 'textures' y 'food': 4577
```

Kmeans

```
from sklearn.cluster import KMeans
```

```
# Desenrollar las listas de embeddings correctamente
#embeddings_flat = np.vstack(conjunto_de_datos['embeddings_ELMo'].apply(np.vstack))
from sklearn.cluster import KMeans
# Desenrollar las listas de embeddings correctamente
embeddings_flat = np.vstack(conjunto_de_datos['Word Embeddings'].apply(np.vstack))

# Número de clústeres
k = 6

# Inicializar y ajustar el modelo KMeans
kmeans = KMeans(n_clusters=k)
kmeans.fit(embeddings_flat)

# Obtener las etiquetas de clúster por cada conjunto único de embeddings
etiquetas_por_fila = kmeans.predict(embeddings_flat)

# Asegurarse de obtener solo dos etiquetas (una por fila)
etiquetas_por_fila = etiquetas_por_fila[:conjunto_de_datos.shape[0]]

# Imprimir la forma de tus datos y las etiquetas para depurar
print("Shape of conjunto_de_datos:", conjunto_de_datos.shape)
print("Shape of etiquetas_por_fila:", etiquetas_por_fila.shape)

# Asignar las etiquetas al DataFrame
conjunto_de_datos['cluster_label'] = etiquetas_por_fila
display(conjunto_de_datos)
# Ahora, tu DataFrame tiene una nueva columna 'cluster_label' con las etiquetas de clúster asignadas por KMeans
```

```
  warnings.warn(
Shape of conjunto_de_datos: (50000, 13)
Shape of etiquetas_por_fila: (50000,)
```

Exploracion de Datos

Thank you thank much

```
display(conjunto_de_datos)
```

	body	id	score	created_utc	created_datetime	Cleaned Body
12	Concerta. It works well for me. Tried focalin ...	ddm0yq2	2	1486818945	2017-02-11 13:15:45	concerta works well tried focalin ritalin swit...
77	Probably both. At least that's what I noticed...	dk8hzhv	2	1500076674	2017-07-14 23:57:54	probably least noticed
78	Yeah I agree probably both. I do notice it in ...	dk8kdjg	2	1500080248	2017-07-15 00:57:28	yeah agree probably notice morning bit drinkin...
92	Honestly all my life I have felt different &am...	dkafl9r	1	1500208332	2017-07-16 12:32:12	honestly life felt different amp know frustrat...
115	Im 29, had my ovaries removed 1 month ago... i...	dl02bo2	1	1501567320	2017-08-01 06:02:00	im ovaries removed month ago also adhd im defi...
...	...	...	...	...	...	...
202581	I am more watching replies than	gwubby4	2	1620086023	2021-05-03 23:53:43	watching replies anything say

```
# Filtra las filas donde 'emocion_predominante' no es igual a 'empty'
conjunto_de_datos_filtrado = conjunto_de_datos.loc[conjunto_de_datos['emocion_predominante'] != 'empty']
conjunto_de_datos = conjunto_de_datos_filtrado
display(conjunto_de_datos)
# Ahora 'conjunto_de_datos_filtrado' contiene solo las filas donde 'emocion_predominante' no es 'empty'.
```

My psych psych told [psy

```
# Seleccionar las columnas id, Cleaned Body y emocion_predominante
diccionario_tdah = conjunto_de_datos[['id', 'Cleaned Body', 'emocion_predominante']]

# Renombrar las columnas
diccionario_tdah = diccionario_tdah.rename(columns={'id': 'id', 'Cleaned Body': 'content', 'emocion_predominante': 'sentime
diccionario_tdah['content'] = diccionario_tdah['content'].str.lower()
display(diccionario_tdah)
```

Enviar Archivo

```
# Supongamos que 'carteraSantander' es tu DataFrame

# Especifica el nombre del archivo de Excel y la hoja donde deseas guardar los datos
nombre_archivo = "/content/drive/MyDrive/Datas/Comentarios_clustering2labelvector50.xlsx"
nombre_hoja = "Cartera"  # Puedes cambiar el nombre de la hoja si lo deseas

# Guarda el DataFrame en un archivo de Excel
conjunto_de_datos.to_excel(nombre_archivo, sheet_name=nombre_hoja, index=False)

print(f"DataFrame guardado en {nombre_archivo}, en la hoja '{nombre_hoja}'.")
```

```
DataFrame guardado en /content/drive/MyDrive/Datas/Comentarios_clustering2labelvector50.xlsx, en la hoja 'Cartera'.
```

Prediccion Modelo

```
import pandas as pd
from sklearn.model_selection import train_test_split

# Dividir el DataFrame en conjuntos de entrenamiento y prueba
train_df, test_df = train_test_split(conjunto_de_datos_equilibrado, test_size=0.2, random_state=42)

# train_df contendrá el conjunto de entrenamiento
# test_df contendrá el conjunto de prueba


from sklearn.ensemble import RandomForestClassifier



from sklearn.feature_extraction.text import TfidfVectorizer

# Crear un objeto TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=1000)  # Puedes ajustar el número de características según tus necesidades

# Ajustar y transformar los textos vectorizados en el conjunto de entrenamiento
X_train = tfidf_vectorizer.fit_transform(train_df['Cleaned Body'])

# Transformar los textos vectorizados en el conjunto de prueba
X_test = tfidf_vectorizer.transform(test_df['Cleaned Body'])

# Ahora puedes ajustar el modelo utilizando X_train y evaluarlo en X_test
model = RandomForestClassifier()
model.fit(X_train, train_df['emocion_predominante'])

# Realizar predicciones en el conjunto de prueba
predictions = model.predict(X_test)
```