Mi código antes de subir a Github

# Importar las bibliotecas necesarias

import pandas as pd

import json

import ast

import csv

import os

from nltk.sentiment.vader import SentimentIntensityAnalyzer

import numpy as np

from scipy import stats

# Cargar los datos de juegos desde un archivo JSON

steam\_games\_data = []

with open('steam\_games.json', 'r', encoding='utf-8') as file:

for line in file:

game\_data = json.loads(line)

steam\_games\_data.append(game\_data)

steam\_games\_df = pd.DataFrame(steam\_games\_data)

# Seleccionar columnas relevantes para tu proyecto

columnas\_relevantes = ['id', 'app\_name', 'genres', 'tags', 'developer', 'price', 'release\_date', 'specs', 'early\_access']

steam\_games\_relevantes\_df = steam\_games\_df[columnas\_relevantes]

# Crear una copia de la columna 'genres'

steam\_games\_relevantes\_df['genres\_cleaned'] = steam\_games\_relevantes\_df['genres'].apply(lambda x: ', '.join(x) if isinstance(x, list) else x)

# Convertir las listas de 'tags' en cadenas separadas por comas

steam\_games\_relevantes\_df['tags\_cleaned'] = steam\_games\_relevantes\_df['tags'].apply(lambda x: ', '.join(x) if isinstance(x, list) else x)

# Codificación one-hot para 'tags\_cleaned'

steam\_games\_relevantes\_df = pd.get\_dummies(steam\_games\_relevantes\_df, columns=['tags\_cleaned'])

# Crear una columna 'release\_year' a partir de 'release\_date'

steam\_games\_relevantes\_df['release\_year'] = pd.to\_datetime(steam\_games\_relevantes\_df['release\_date'], errors='coerce').dt.year

steam\_games\_relevantes\_df['release\_year'].fillna(0, inplace=True)

steam\_games\_relevantes\_df['release\_year'] = steam\_games\_relevantes\_df['release\_year'].astype(int)

# Guardar el DataFrame con los cambios en un archivo CSV

steam\_games\_relevantes\_df.to\_csv('steam\_games\_relevantes.csv', index=False)

# Cargar léxico VADER desde un archivo CSV

vader\_lexicon\_csv = os.path.abspath('vader\_lexicon.csv')

vader\_lexicon = {}

with open(vader\_lexicon\_csv, newline='', encoding='utf-8') as csvfile:

csvreader = csv.reader(csvfile, delimiter=',')

for row in csvreader:

if len(row) == 3:

word, sentiment, \_ = row

vader\_lexicon[word] = float(sentiment)

# Cargar datos de usuarios e historiales de revisiones

rows\_users\_items = []

with open('users\_items.json', encoding='utf-8') as f:

for line in f.readlines():

rows\_users\_items.append(ast.literal\_eval(line))

users\_items\_df = pd.DataFrame(rows\_users\_items)

rows\_user\_reviews = []

with open('user\_reviews.json', encoding='utf-8') as f:

for line in f.readlines():

rows\_user\_reviews.append(ast.literal\_eval(line))

user\_reviews\_df = pd.DataFrame(rows\_user\_reviews)

# Eliminar columnas innecesarias en users\_items\_df

columns\_to\_keep = ['user\_id', 'items\_count', 'steam\_id', 'user\_url']

users\_items\_df = users\_items\_df[columns\_to\_keep]

# Crear una instancia de SentimentIntensityAnalyzer

analyzer = SentimentIntensityAnalyzer()

# Crear una función para obtener el puntaje de sentimiento

def get\_sentiment\_score(sentiment\_list):

if sentiment\_list is None:

return 0.0

sentiment\_text = ' '.join(map(str, sentiment\_list))

sentiment = analyzer.polarity\_scores(sentiment\_text)

return sentiment['compound']

# Aplicar la función a la columna 'reviews' y almacenar los resultados en una nueva columna 'sentiment\_score'

user\_reviews\_df['sentiment\_score'] = user\_reviews\_df['reviews'].apply(get\_sentiment\_score)

# Filtrar las revisiones con puntaje de sentimiento positivo

positive\_reviews = user\_reviews\_df[user\_reviews\_df['sentiment\_score'] > 0]

# Filtrar las revisiones con puntaje de sentimiento negativo

negative\_reviews = user\_reviews\_df[user\_reviews\_df['sentiment\_score'] < 0]

# Guardar los DataFrames de revisiones positivas y negativas en archivos CSV

positive\_reviews.to\_csv('positive\_reviews.csv', index=False)

negative\_reviews.to\_csv('negative\_reviews.csv', index=False)

# Crear un nuevo DataFrame con el análisis de sentimientos

sentiment\_analysis\_df = pd.DataFrame()

# Calcular estadísticas de las revisiones positivas

positive\_stats = positive\_reviews['sentiment\_score'].describe()

# Calcular estadísticas de las revisiones negativas

negative\_stats = negative\_reviews['sentiment\_score'].describe()

# Agregar estadísticas al DataFrame de análisis de sentimientos

sentiment\_analysis\_df['Positive Reviews'] = positive\_stats

sentiment\_analysis\_df['Negative Reviews'] = negative\_stats

# Guardar el DataFrame de análisis de sentimientos en un archivo CSV

sentiment\_analysis\_df.to\_csv('sentiment\_analysis.csv', index=False)