Ibrahim Garcia Ibañez

Materia: Introducción a la ciencia de datos

Profesor: JAIME ALEJANDRO ROMERO SIERRA

Fecha de entrega: 20/10/25

https://github.com/ibrahimelCABALLOesquizofrenico/ProyectoLimpiezaDeBase

INDICE

Descripicion del proyecto

Contexto de la base de datos

El éxito o la popularidad de un videojuego depende de una combinación de factores internos y externos. Entre ellos, el precio del juego influye en la decisión de compra, mientras que las calificaciones y reseñas, tanto positivas como negativas, afectan la percepción del público y su disposición a recomendarlo. El género del juego y los modos multijugador también juegan un papel importante, ya que determinan el tipo y tamaño de la audiencia, así como la actividad de la comunidad. Además, la fecha de lanzamiento puede ser determinante, evitando competencia o aprovechando temporadas clave, mientras que estrategias de marketing y promoción incrementan su visibilidad. Factores internos como la jugabilidad, historia, gráficos y música impactan indirectamente al influir en la satisfacción de los jugadores. Finalmente, tendencias culturales y la exposición en redes y plataformas de streaming pueden catapultar la popularidad de un título más allá de sus características técnicas, haciendo que el éxito sea el resultado de la interacción de múltiples variables a lo largo del tiempo.

Descripicion general del contenido en la base de datos

se encuentran titulos desde el año 2013 hasta el año 2023, su fecha de lanzamiento, su precio normal en steam, su id en steam,su calificacion postiva y negativo, el minimo y el maximo de dueños de un juego y la duracion en horas de la campaña en solitario de un videojuego

significado de cada columna

name(nombre del videojuego)

release_date(cuando salio el juego)

price(precio del videojuego)

positive(calificacion positiva del juego)

negative(calificacion negativa del juego)

app_id(numero que representa al juego)

min_owners(cantidad minima de dueños que tiene el juego)

max_owners(cantidad maxima de dueños que tuvo el juego)

hltb_single(la duracion en horas mde la campaña en solitario (es aproximado))

Columna	Descripción
`name`	Nombre del videojuego
`release_date`	Fecha de lanzamiento
`price`	Precio del videojuego en Steam
`positive`	Número de reseñas positivas
`negative`	Número de reseñas negativas
`app_id`	Identificador único del juego en Steam
`min_owners`	Estimación mínima de propietarios del juego
`max_owners`	Estimación máxima de propietarios del juego
`hltb_single`	Duración aproximada de la campaña en solitario (según HowLongToBeat)

Inicio de la limpieza de la base de datos

```
df_sucio.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 67420 entries, 0 to 67419
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   name          67420 non-null  object
 1   release_date  67420 non-null  object
 2   price         64049 non-null  float64
 3   positive      64049 non-null  float64
 4   negative      62759 non-null  float64
 5   app_id        64049 non-null  float64
 6   min_owners    62762 non-null  float64
 7   max_owners    64049 non-null  object
 8   hltb_single   13616 non-null  float64
dtypes: float64(6), object(3)
memory usage: 4.6+ MB
```

El primer paso fue ver cuántos datos nan y cuantos datos basura habían,también ver cuantas filas se duplicaron. Todo esto se hacía por medio de .info, .isnull, .shape y .duplicated

```
df_sucio.duplicated().sum()
```

```
0.0s
np.int64(6313)
```

```
df_sucio.isnull().sum()
```

```
name               0
release_date       0
price           3371
positive        3371
negative        4661
app_id          3371
min_owners      4658
max_owners      3371
hltb_single    53804
dtype: int64
```

```
df_sucio.shape
```

```
(77628, 9)
```

Código para la limpieza de la base

Codigo para remplazar las palabras basura,los nan y celdas vacias con ceros y eliminar filas y columnas repetidas

```python
#PD: Aqui si use IA profe, para que el codigo funcionara bien
# --- Cargar el CSV sucio ---
df_limpio = pd.read_csv("df_sucio.csv")

# --- Reemplazar palabras basura ---
palabras_basura = ["bbb", "invalid", "Auto%#", "nan", "None", "NULL"]
df_limpio.replace(palabras_basura, 0, inplace=True)

# --- Reemplazar celdas vacías o espacios en blanco con 0 ---
df_limpio.replace(r'^\s*$', 0, regex=True, inplace=True)

# ---Reemplazar valores NaN con 0 ---
df_limpio.fillna(0, inplace=True)

# --- Eliminar filas y columnas completamente vacías ---
df_limpio.dropna(axis=0, how='all', inplace=True)  # Eliminar filas vacías
df_limpio.dropna(axis=1, how='all', inplace=True)  # Eliminar columnas vacías

# --- Eliminar columnas duplicadas ---
df_limpio = df.T.drop_duplicates().T
```

Primero es remplazar las palabras basura con ceros usando un diccionario y el comando .replace

Después se remplazan las celdas vacías por ceros con un comando utilizando la biblioteca regrex y el comando .inplace

Después se eliminan las filas repetidas con dropnas que eliminan las filas y columnas repetidas

Después para eliminar todo completamente utilizo un dropna para eliminar columnas repetidas

Después de hace lo mismo con las filas y ya al final se guarda todo en un mismo csv

```python
# --- Eliminar filas duplicadas ---
df_limpio.drop_duplicates(keep="first", inplace=True)

# --- Guardar el resultado limpio ---
df_limpio.to_csv("df_limpio.csv", index=False, encoding="utf-8-sig")

# --- Mostrar resumen ---
print("Limpieza completada exitosamente.")
print(f"Filas finales: {df_limpio.shape[0]}")
print(f"Columnas finales: {df_limpio.shape[1]}")
print("Archivo guardado como 'df_limpio.csv'")
```

Resultados finales

```
--- Vista previa del DataFrame limpio ---
                    name  release_date  price  positive  negative   app_id  \
0            Train Bandit  Oct 12, 2017   0.99      53.0       5.0   655370
1                 Henosis™  Jul 23, 2020   5.99       3.0       0.0  1355720
2  Two Weeks in Painland   Feb 3, 2020   0.00      50.0       8.0  1139950
3          Wartune Reborn  Feb 26, 2021   0.00      87.0      49.0        0
4               TD Worlds   Jan 9, 2022  10.99      21.0       7.0  1659180

   min_owners  max_owners  hltb_single
0           0     20000.0          0.0
1           0     20000.0          0.0
2           0         0.0          0.0
3       50000    100000.0          0.0
4           0     20000.0          0.0

--- Comprobación de NaNs ---
name            0
release_date    0
price           0
positive        0
negative        0
app_id          0
min_owners      0
max_owners      0
hltb_single     0
dtype: int64
```

Resultado final

```
df_limpio.isnull().sum()
```

Python

```
name            0
release_date    0
price           0
positive        0
negative        0
app_id          0
min_owners      0
max_owners      0
hltb_single     0
dtype: int64
```

```
df_limpio.info()
```

[36] Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 77628 entries, 0 to 77627
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   name          77628 non-null  object
 1   release_date  77628 non-null  object
 2   price         77628 non-null  float64
 3   positive      77628 non-null  float64
 4   negative      77628 non-null  float64
 5   app_id        77628 non-null  int64
 6   min_owners    77628 non-null  int64
 7   max_owners    77628 non-null  float64
 8   hltb_single   77628 non-null  float64
dtypes: float64(5), int64(2), object(2)
memory usage: 5.3+ MB
```

```
df_limpio.shape
```

[37] Python

```
(77628, 9)
```

```python
import random
# --- Función para ensuciar el DataFrame ---
def ensuciar_df(df_original, columnas_protegidas=None):
    df = df_original.copy()

    # Columnas que no deben ensuciarse
    if columnas_protegidas is None:
        columnas_protegidas = ['name', 'release_date']

    # --- Duplicar algunas filas ---
    num_duplicados = random.randint(1, int(len(df) * 0.3))
    duplicadas = df.sample(n=num_duplicados, replace=True)
    df = pd.concat([df, duplicadas], ignore_index=True)

    # --- Insertar valores 'bbb' en columnas no protegidas ---
    num_invalid = int(len(df) * 0.02)
    columnas_modificables = [c for c in df.columns if c not in columnas_protegidas]

    for col in random.sample(columnas_modificables, len(columnas_modificables) // 2):
        if df[col].dtype != 'object':
            df[col] = df[col].astype('object')
        valid_indices = df.index[df[col].notna()]
        if len(valid_indices) > 0:
            invalid_indices = np.random.choice(valid_indices, size=min(num_invalid, len(valid_indices)), replace=False)
            df.loc[invalid_indices, col] = 'bbb'
```

Primero es definir otra variable la cual hará que la base original la ensuciara denuveo y después guardara los cambios y los pasara en un nuevo csv llamado df_sucio.csv

```python
    # --- Cambiar tipos de formato en columnas no protegidas ---
    columnas_a_cambiar = random.sample(columnas_modificables, random.randint(1, len(columnas_modificables) // 2))
    for col in columnas_a_cambiar:
        if pd.api.types.is_numeric_dtype(df[col]):
            df[col] = df[col].astype(str)
        else:
            df[col] = pd.to_numeric(df[col], errors='coerce')

    # --- Insertar NaNs en un 5% de las celdas, excepto las protegidas ---
    num_nan = int(len(df) * 0.05)
    for col in columnas_modificables:
        nan_indices = np.random.choice(df.index, size=min(num_nan, len(df)), replace=False)
        df.loc[nan_indices, col] = np.nan

    return df


# --- Cargar el CSV original ---
ruta_csv = 'https://raw.githubusercontent.com/ibrahimelCABALLOesquizofrenico/Proyecto-cienciasdedatos/refs/heads/main/games%20steam%202013-2023.csv?token=GHSAT
df = pd.read_csv(ruta_csv)

# --- Aplicar el ensuciado ---
df_sucio = ensuciar_df(df, columnas_protegidas=['name', 'release_date'])

# --- Guardar el DataFrame sucio en un nuevo archivo CSV ---
df_sucio.to_csv('df_sucio.csv', index=False, encoding='utf-8-sig')
```

[Es

Aplicamos casi lo mismo para la base limpia pero en vez de ensuciar la base las limpiamos de nuevo y guardamos los cambios en un archivo llamado df_limpio.csv

```
Base de datos sucia creada exitosamente como 'df_sucio.csv'
```

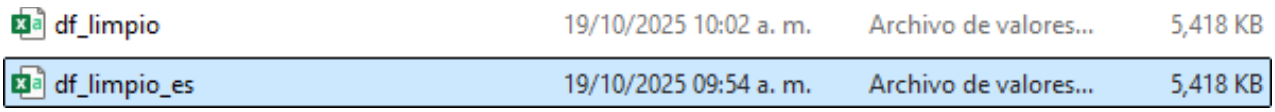
```python
# --- Funcion para limpiar el DataFrame ---
def limpiar_df(df):
    df_limpio = df.copy()

    # Reemplazar los valores "bbb" (o cualquier otro texto basura) por NaN
    df_limpio = df_limpio.replace('bbb', np.nan)

    # Reemplazar los NaN con 0
    df_limpio = df_limpio.fillna(0)

    # Asegurar que las columnas numéricas vuelvan a ser tipo numérico cuando sea posible
    for col in df_limpio.columns:
        df_limpio[col] = pd.to_numeric(df_limpio[col], errors='ignore')

    return df_limpio


# --- Cargar el CSV sucio ---
df_sucio = pd.read_csv('df_sucio.csv')

# --- Aplicar la limpieza ---
df_limpio = limpiar_df(df_sucio)

# --- Guardar el DataFrame limpio ---
df_limpio.to_csv('df_limpio.csv', index=False, encoding='utf-8-sig')
```

```
Base de datos limpia creada exitosamente como 'df_limpio.csv
```

Traducción de columnas

Primero hacemos un diccionario con todas las columnas traducidas después definimos un variable la cual remplazara los nombres de las columnas con sus versiones traducidas y de ahí sacar nuevos archivos con las extencion .es

EJEMPLO



```python
# --- Diccionario de traducción ---
traducciones = {
    "name": "nombre",
    "release_date": "fecha_de_lanzamiento",
    "price": "precio",
    "positive": "calificacion_positivas",
    "negative": "calificacion_negativas",
    "app_id": "id_del_juego",
    "min_owners": "minimo_de_dueños",
    "max_owners": "maximo_de_dueños",
    "hltb_single": "duracion_de_la_campaña_aprox",
    }

# --- Función para traducir columnas ---
def traducir_columnas(df, diccionario):
    nuevos_nombres = {col: diccionario.get(col, col) for col in df.columns}
    df.rename(columns=nuevos_nombres, inplace=True)
    return df, nuevos_nombres

# --- Traducir el csv sucio ---
try:
    df_sucio = pd.read_csv("df_sucio.csv", encoding="utf-8")
    df_sucio, nombres_sucio = traducir_columnas(df_sucio, traducciones)
    df_sucio.to_csv("df_sucio_es.csv", index=False, encoding="utf-8-sig")
    print(" Archivo 'df_sucio.csv' traducido y guardado como 'df_sucio_es.csv'")
except FileNotFoundError:
    print("No se encontró el archivo 'df_sucio.csv'. Asegúrate de tenerlo en la misma carpeta.")
```

```python
# --- Traducir el csv limpio ---
try:
    df_limpio = pd.read_csv("df_limpio.csv", encoding="utf-8")
    df_limpio, nombres_limpio = traducir_columnas(df_limpio, traducciones)
    df_limpio.to_csv("df_limpio_es.csv", index=False, encoding="utf-8-sig")
    print(" Archivo 'df_limpio.csv' traducido y guardado como 'df_limpio_es.csv'")
except FileNotFoundError:
    print(" No se encontró el archivo 'df_limpio.csv'. Asegúrate de tenerlo en la misma carpeta.")
```

```
 Archivo 'df_sucio.csv' traducido y guardado como 'df_sucio_es.csv'
 Archivo 'df_limpio.csv' traducido y guardado como 'df_limpio_es.csv'

Traducciones aplicadas:
name → nombre
release_date → fecha_de_lanzamiento
price → precio
positive → calificacion_positivas
negative → calificacion_negativas
app_id → id_del_juego
min_owners → minimo_de_dueños
max_owners → maximo_de_dueños
hltb_single → duracion_de_la_campaña_aprox
```

Conclusiones

1. Problemas que presento la base

No presento muchos problemas ya que la base era fácil de manejar y no tenía tantos valores atípicos aparte que las columnas tenían valores específicos que no se pueden repetir

Problemas Detectados

- **Presencia de valores nulos en múltiples columnas**
- **Palabras basura en campos de texto**
- **Tipos de datos inconsistentes**

2. ¿Qué técnicas aplicaste para solucionarlos?

Primero ver cuales eran las palabras basuras e identificar donde había más nan y de ahí aplicar el comando de fillna con el
remplazar palabras basuras con un

cero Técnicas Aplicadas

- **Reemplazo de valores nulos con ceros**
- **Eliminación de palabras basura**
- **Conversión de tipos de datos**

- **Traducción de columna**

3. ¿Que aprendí?

A como funciona un proyecto de limpieza y de cómo funciona mucho el hecho de tener una lista de comandos extra para la limpieza de los datos

Aprendizajes

- **Comprensión del flujo completo de limpieza de datos**
- **Importancia de validar y transformar datos antes del análisis**
- **Uso de funciones personalizadas para automatizar proceso**