c9422a73-b3ae-49fc-a64f-3d6d61cab188

November 26, 2024

¡Hola, Jafet!

Mi nombre es Tonatiuh Cruz. Me complace revisar tu proyecto hoy.

Al identificar cualquier error inicialmente, simplemente los destacaré. Te animo a localizar y abordar los problemas de forma independiente como parte de tu preparación para un rol como data-analyst. En un entorno profesional, tu líder de equipo seguiría un enfoque similar. Si encuentras la tarea desafiante, proporcionaré una pista más específica en la próxima iteración.

Encontrarás mis comentarios a continuación - **por favor no los muevas, modifiques o elimines**.

Puedes encontrar mis comentarios en cajas verdes, amarillas o rojas como esta:

Comentario del revisor

Éxito. Todo está hecho correctamente.

Comentario del revisor

Observaciones. Algunas recomendaciones.

Comentario del revisor

Necesita corrección. El bloque requiere algunas correcciones. El trabajo no puede ser aceptado con comentarios en rojo.

Puedes responderme utilizando esto:

Resumen de la revisión 1

Hola Jafet! Has hecho un excelente trabajo con los ajustes, cada vez más cercas de convertirte en un analista de datos. Realizaste la carga de bases, su análisis inicial, sus ajustes, un análisis de algunos variables conmplementado con gráficas y el desarrollo de las pruebas de hipótesis. Solamente te dejo algunos comentarios para complementar el análisis.

Sigue con el excelente trabajo!

Resumen de la revisión 2

Hola Jafet! Has hecho un excelente trabajo con los ajustes, completaste los valores nulos y desarrollaste la prueba de levene para mostrar el supuesto de varianzas iguales.

Sigue con el excelente trabajo!

[5]:
```
# PASO 1
```

```
import pandas as pd

# Cargar el archivo CSV
games_df = pd.read_csv('/datasets/games.csv')
print("Archivo cargado exitosamente con read_csv:\n")
games_df.head()
```

```
Archivo cargado exitosamente con read_csv:
```

[5]:

	Name	Platform	Year_of_Release	Genre	NA_sales	EU_sales	JP_sales	Other_sales	Critic_Score	User_Score	Rating
0	Wii Sports	Wii	2006.0	Sports	41.36	28.96	3.77	8.45	76.0	8	E
1	Super Mario Bros.	NES	1985.0	Platform	29.08	3.58	6.81	0.77	NaN	NaN	NaN
2	Mario Kart Wii	Wii	2008.0	Racing	15.68	12.76	3.79	3.29	82.0	8.3	E
3	Wii Sports Resort	Wii	2009.0	Sports	15.61	10.93	3.28	2.95	80.0	8	E
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	11.27	8.89	10.22	1.00	NaN	NaN	NaN

Comentario del revisor:

Has realizado un excelente trabajo al importar los datos y las bibliotecas necesarias.

[8]:
```
# PASO 2

# Convertir los nombres de las columnas a minúsculas
games_df.columns = games_df.columns.str.lower()
print("Nombres de las columnas después de cambiar a minúsculas:\n", games_df.
 ↪columns)

# Convertir los datos a los tipos necesarios
# Revisemos los tipos actuales de las columnas
print("\nTipos de datos originales:\n", games_df.dtypes)


# Explicacion:
# year_of_release: Convertimos los valores no válidos (TBD) en NaN, y luego␣
 ↪cambiamos a un tipo Int64 que permite valores nulos.
# user_score: Convertimos TBD a NaN y cambiamos los valores válidos a float.


# Columna 'year_of_release': asegurarse de que esté en formato entero, pero␣
 ↪primero tratemos valores nulos o 'TBD'
```

```
# Reemplazamos 'TBD' en 'year_of_release' y convertimos a int
games_df['year_of_release'] = pd.to_numeric(games_df['year_of_release'],␣
 ↪errors='coerce').astype('Int64')

# Columna 'user_score': algunos valores podrían estar como 'TBD'
games_df['user_score'] = pd.to_numeric(games_df['user_score'], errors='coerce')

# Verificamos los cambios
print("\nTipos de datos después de las conversiones:\n", games_df.dtypes)

# Tratar valores ausentes
# Revisemos los valores ausentes en el DataFrame
missing_values = games_df.isna().sum()
print("\nValores ausentes en cada columna:\n", missing_values)

# Decisiones sobre valores ausentes:
# - year_of_release: podríamos optar por dejar en blanco si es un valor␣
 ↪importante para el análisis.
# - user_score: TBD convertido a NaN puede quedarse como tal, o podríamos␣
 ↪decidir rellenarlo si hay sentido hacerlo.

# En este caso, dejaremos los valores ausentes en year_of_release y user_score
# Explicación de por qué están ausentes: podría ser que aún no se tenían estos␣
 ↪datos al momento de crear el dataset.

# Calcular las ventas totales
# Sumamos las ventas en las regiones para obtener la venta total
sales_columns = ['na_sales', 'eu_sales', 'jp_sales', 'other_sales']
games_df['total_sales'] = games_df[sales_columns].sum(axis=1)


# Completar valores faltantes de 'user_score' con la mediana
user_score_median = games_df['user_score'].median()
games_df['user_score'] = games_df['user_score'].fillna(user_score_median)


# Verificamos el resultado final
print("\nDataFrame final con columna de ventas totales:\n")
games_df.head()
```

```
Nombres de las columnas después de cambiar a minúsculas:
 Index(['name', 'platform', 'year_of_release', 'genre', 'na_sales', 'eu_sales',
       'jp_sales', 'other_sales', 'critic_score', 'user_score', 'rating',
       'total_sales'],
      dtype='object')

Tipos de datos originales:
```

```
 name                object
platform            object
year_of_release      Int64
genre               object
na_sales           float64
eu_sales           float64
jp_sales           float64
other_sales        float64
critic_score       float64
user_score         float64
rating              object
total_sales        float64
dtype: object

Tipos de datos después de las conversiones:
 name                object
platform            object
year_of_release      Int64
genre               object
na_sales           float64
eu_sales           float64
jp_sales           float64
other_sales        float64
critic_score       float64
user_score         float64
rating              object
total_sales        float64
dtype: object

Valores ausentes en cada columna:
 name                 2
platform              0
year_of_release     269
genre                 2
na_sales              0
eu_sales              0
jp_sales              0
other_sales           0
critic_score       8578
user_score         9125
rating             6766
total_sales           0
dtype: int64

DataFrame final con columna de ventas totales:
```

[8]:
```
                      name platform  year_of_release         genre  na_sales  \
0               Wii Sports      Wii             2006        Sports     41.36
1        Super Mario Bros.      NES             1985      Platform     29.08
2           Mario Kart Wii      Wii             2008        Racing     15.68
3        Wii Sports Resort      Wii             2009        Sports     15.61
4  Pokemon Red/Pokemon Blue       GB             1996  Role-Playing     11.27

   eu_sales  jp_sales  other_sales  critic_score  user_score rating  \
0     28.96      3.77         8.45          76.0         8.0      E
1      3.58      6.81         0.77           NaN         7.5    NaN
2     12.76      3.79         3.29          82.0         8.3      E
3     10.93      3.28         2.95          80.0         8.0      E
4      8.89     10.22         1.00           NaN         7.5    NaN

   total_sales
0        82.54
1        40.24
2        35.52
3        32.77
4        31.38
```

Comentario del revisor:

En ocasiones podemos completar los valores faltantes con información con proxys. Por ejemplo en este caso los podríamos completar con los años de lanzamientos de los juegos en otras plataformas

Comentario del revisor:

Hola! Muy buen trabajo en la sección, ajustaste los nombres de las columnas a minúsculas con el uso de la función str.lower(), cambiaste el tipo de variable de la base de datos y muy buen trabajo con la suma de todas las ventas.

Comentario del revisor:

Solamente para el caso de user_score te recomiendo también completar los valores nulos con la mediana para no sesgar nuestros resultados, esto dado que estamos trabajando con distribuciones sesgadas

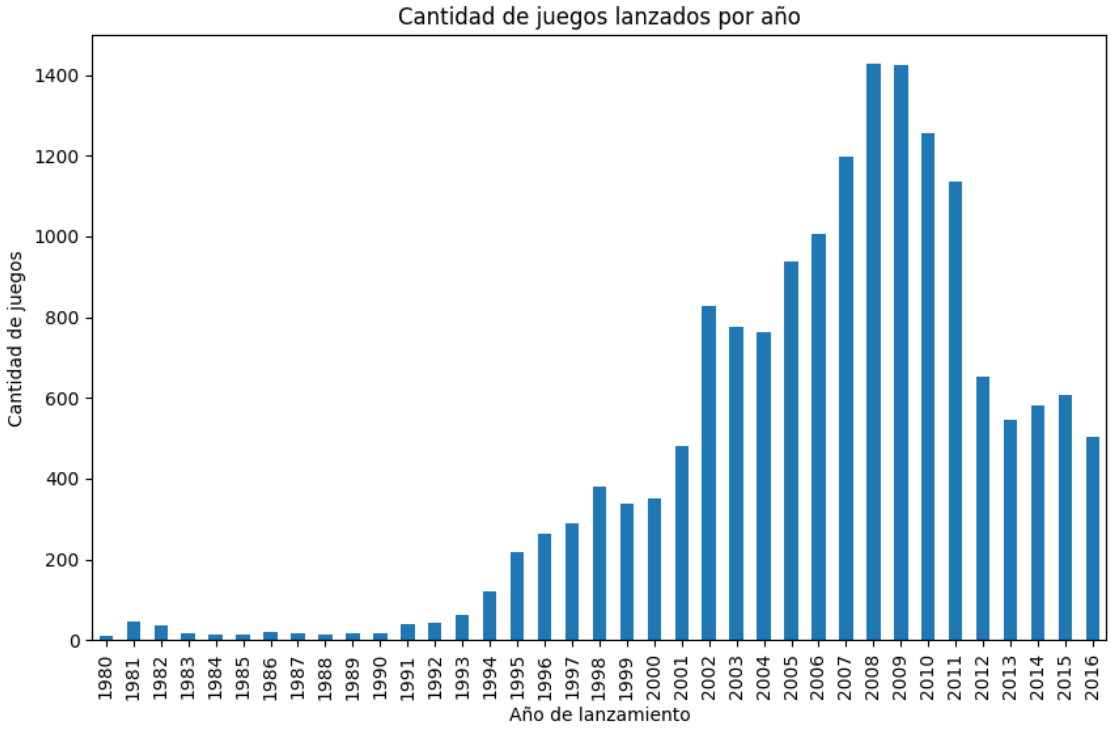
LISTO!

Comentario del revisor:

Gran trabajo completando los valores nulos!

[24]:
```
#Paso 3.1: Juegos lanzados en diferentes años

# Contar la cantidad de juegos lanzados por año
games_per_year = games_df['year_of_release'].value_counts().sort_index()

# Visualización: Juegos lanzados por año
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(10, 6))
games_per_year.plot(kind='bar')
plt.title('Cantidad de juegos lanzados por año')
plt.xlabel('Año de lanzamiento')
plt.ylabel('Cantidad de juegos')
plt.show()

#Interpretación: Esto nos permitirá ver si los datos de cada período son␣
 ↪significativos en cuanto a la cantidad de lanzamientos.
```

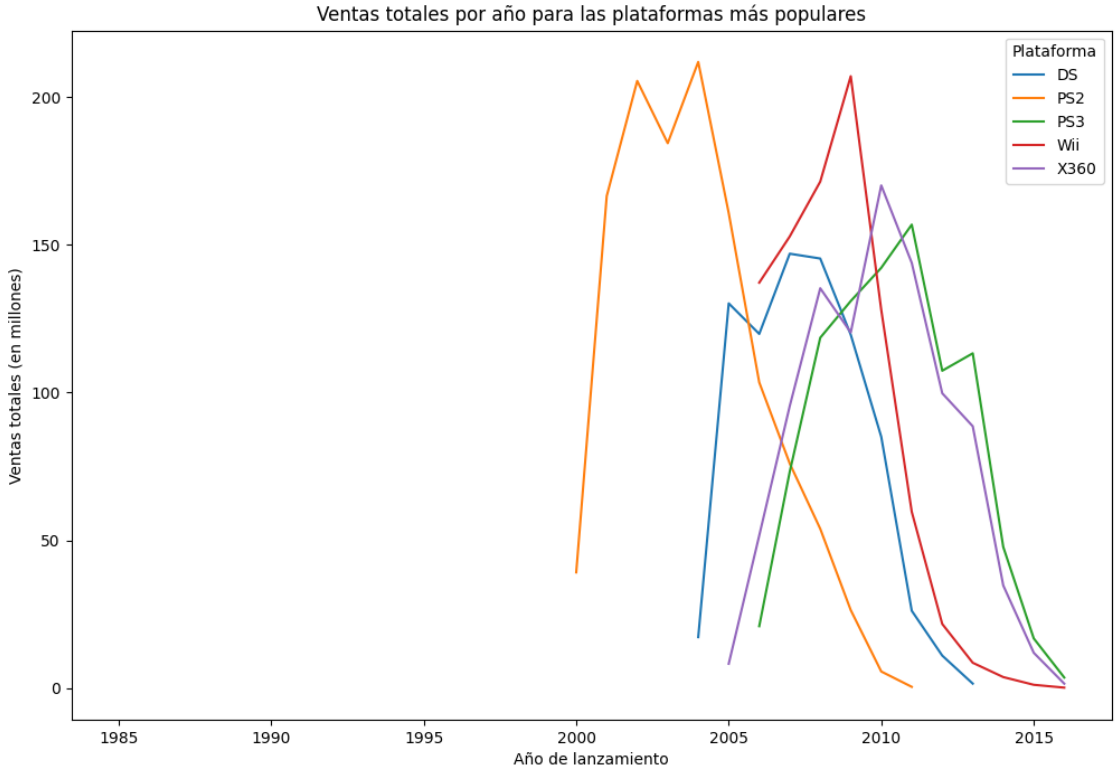


Comentario del revisor:

Gran trabajo con el desarrollo de la gráfica, esta gráfica nos ayuda a entender también sobre el comportamiento del mercado de videojuegos en el tiempo

[10]:
```
###Paso 3.2: Análisis de ventas por plataforma

# Calcular ventas totales por plataforma y seleccionar las plataformas con␣
 ↪mayores ventas
platform_sales = games_df.groupby('platform')['total_sales'].sum().
 ↪sort_values(ascending=False)

# Seleccionar las plataformas más populares
```

```
top_platforms = platform_sales.head(5).index

# Filtrar solo las plataformas principales y observar las ventas anuales
top_platforms_df = games_df[games_df['platform'].isin(top_platforms)]

# Ventas por año para cada plataforma
sales_by_year_platform = top_platforms_df.pivot_table(
    index='year_of_release', columns='platform', values='total_sales',␣
 ↪aggfunc='sum'
)

# Visualización: Ventas de plataformas seleccionadas por año
sales_by_year_platform.plot(figsize=(12, 8))
plt.title('Ventas totales por año para las plataformas más populares')
plt.xlabel('Año de lanzamiento')
plt.ylabel('Ventas totales (en millones)')
plt.legend(title='Plataforma')
plt.show()

#Interpretación: Aquí podemos observar qué plataformas tuvieron un auge de␣
 ↪ventas y cuáles han caído en popularidad, identificando plataformas antiguas␣
 ↪que ya no generan ventas.
```

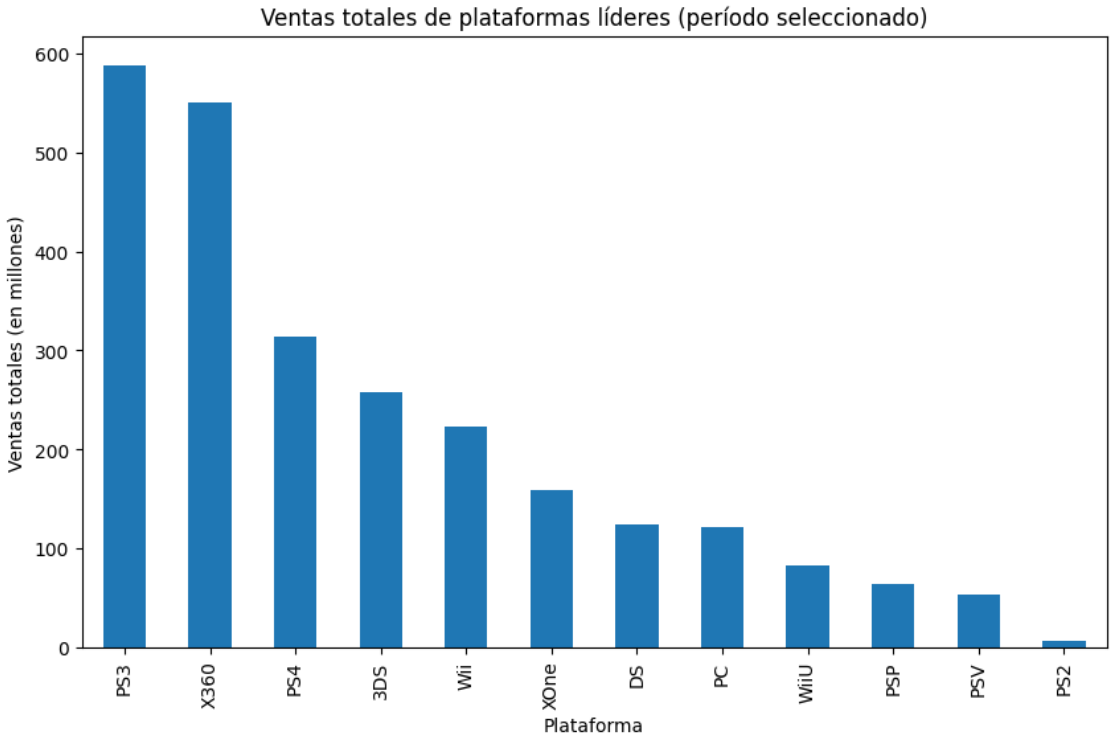


Comentario del revisor

Gran trabajo con el desarrollo de la gráfica. Solamente te recomendaría que intentes reducir el periodo dado que el perido actual puede resultar muy extenso. Recordemos que con base en la información la popularidad la alcanzan en los primeros años. Es por ello que con un periodo largo puede no ser posible analizar de forma clara las plataformas y videojuegos actualizados y podrías estar analizando los desactualizados. Es por ello y dado que buscamos responder qué vender en 2017 te recomendareia gráficar un periodo que no supere los 5 años.

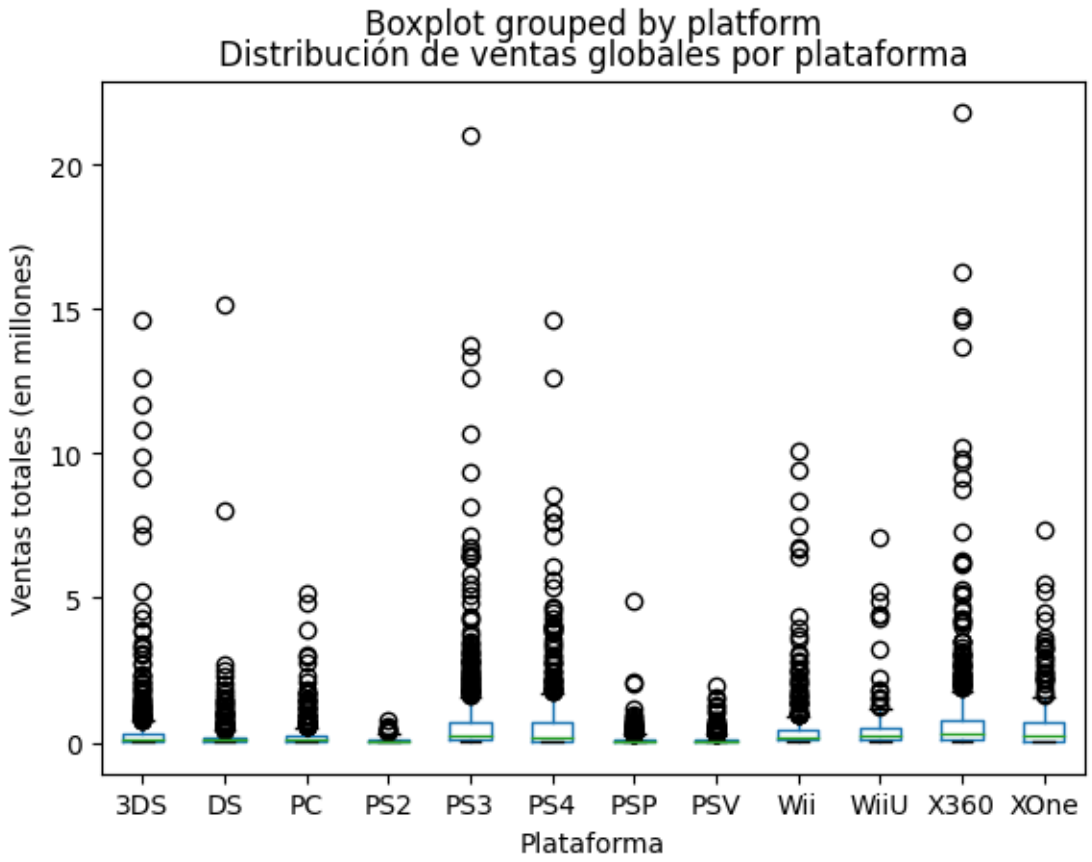
[11]:
```
#Paso 3.3: Período de análisis relevante para el modelo de 2017

# Filtrar los datos del período seleccionado, por ejemplo, desde 2010 en␣
 ↪adelante
relevant_data = games_df[games_df['year_of_release'] >= 2010]
```

[12]:
```
#Paso 3.4: Análisis de crecimiento de plataformas

# Analizar plataformas líderes en ventas en el período seleccionado
leader_platforms_sales = relevant_data.groupby('platform')['total_sales'].sum().
 ↪sort_values(ascending=False)

# Gráfico de las plataformas más rentables
leader_platforms_sales.plot(kind='bar', figsize=(10, 6))
plt.title('Ventas totales de plataformas líderes (período seleccionado)')
plt.xlabel('Plataforma')
plt.ylabel('Ventas totales (en millones)')
plt.show()

#Interpretación: Esto nos ayuda a ver qué plataformas siguen creciendo y cuáles␣
 ↪están en declive, lo que permite identificar posibles plataformas rentables.
```



[13]:
```
#Paso 3.5: Diagrama de caja de ventas globales por plataforma

    # Diagrama de caja para ventas globales por plataforma
plt.figure(figsize=(12, 8))
relevant_data.boxplot(column='total_sales', by='platform', grid=False)
plt.title('Distribución de ventas globales por plataforma')
plt.xlabel('Plataforma')
plt.ylabel('Ventas totales (en millones)')
plt.show()


#Interpretacion: Observaremos diferencias significativas en las ventas por
#plataforma, permitiendo comparar el desempeño de cada una.
```

```
<Figure size 1200x800 with 0 Axes>
```

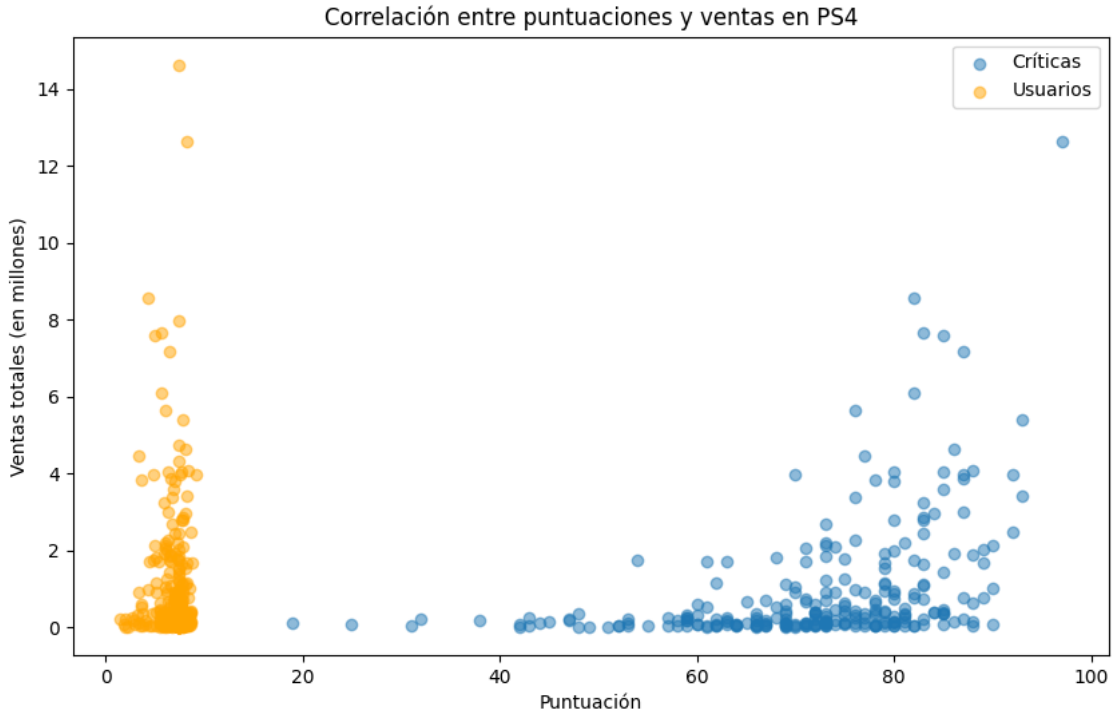


Comentario del revisor

Muy buen trabajo con el desarrollo de la gráfica de cajas, este tipo de gráficos nos ayuda a comparar resultados de manera precisa. Solamente recuerda que los graficos de box-plot nos ayudan a identificar outliers que posteriomente en el desarrollo de las pruebas de hipótesis nos servirían eliminar para que no puedan sesgar nuestros resultados.

[14]:
```
#Paso 3.6: Efecto de las reseñas de usuarios y críticos en las ventas

# Seleccionamos una plataforma popular, por ejemplo, 'PS4'
popular_platform_df = relevant_data[relevant_data['platform'] == 'PS4']

# Crear gráficos de dispersión para analizar correlación
plt.figure(figsize=(10, 6))
plt.scatter(popular_platform_df['critic_score'],␣
 ↪popular_platform_df['total_sales'], alpha=0.5, label="Críticas")
plt.scatter(popular_platform_df['user_score'],␣
 ↪popular_platform_df['total_sales'], alpha=0.5, label="Usuarios",␣
 ↪color="orange")
plt.title('Correlación entre puntuaciones y ventas en PS4')
```

```
plt.xlabel('Puntuación')
plt.ylabel('Ventas totales (en millones)')
plt.legend()
plt.show()

# Calcular correlaciones
critic_correlation = popular_platform_df['critic_score'].
 ↪corr(popular_platform_df['total_sales'])
user_correlation = popular_platform_df['user_score'].
 ↪corr(popular_platform_df['total_sales'])
print(f"Correlación entre crítica y ventas: {critic_correlation}")
print(f"Correlación entre puntuación de usuario y ventas: {user_correlation}")

#Interpretación: Observamos si existe una correlación significativa entre las␣
 ↪puntuaciones y las ventas en esta plataforma.
```



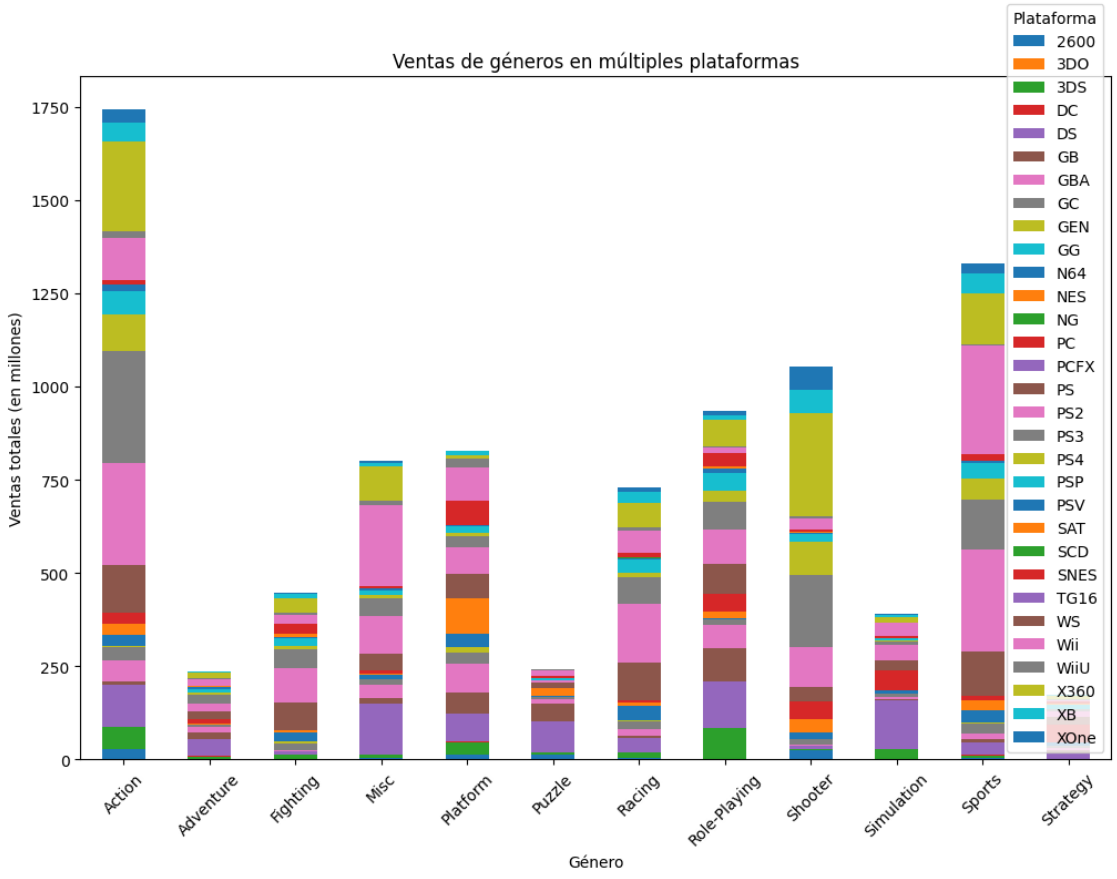

```
Correlación entre crítica y ventas: 0.40656790206178095
Correlación entre puntuación de usuario y ventas: -0.06275557891282788
```

Comentario del revisor

Muy buen trabajo con el análisis de la relación estre las scores y las ventas.

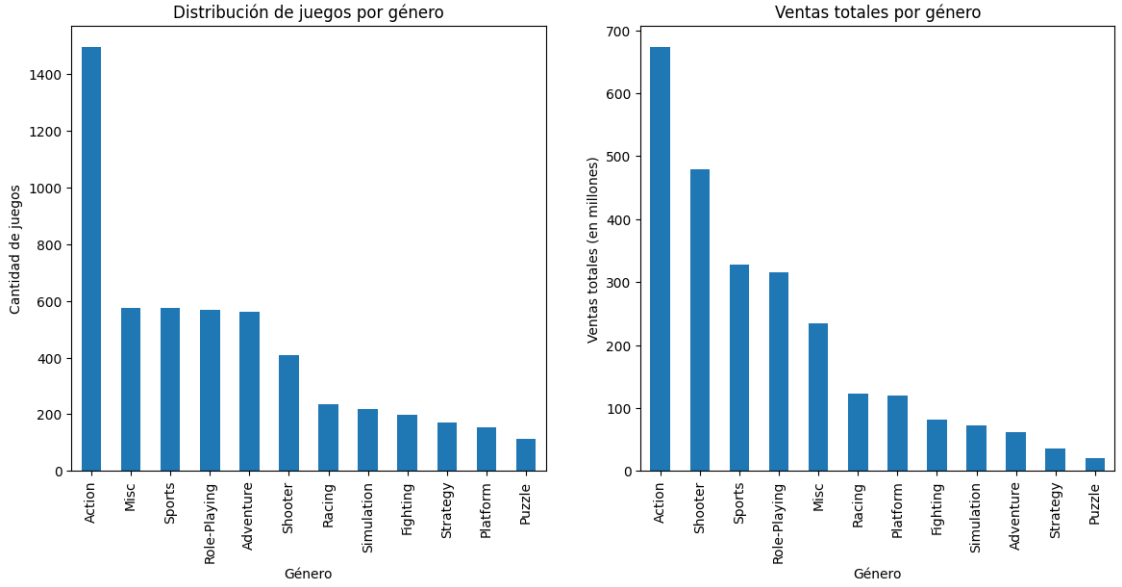
[15]:
```
#Paso 3.7: Comparación de ventas en varias plataformas para los mismos juegos
```

```
# Filtrar géneros que aparecen en varias plataformas
multi_genre_platforms = games_df[games_df.duplicated('genre', keep=False)]

# Analizar las ventas en cada plataforma por género
multi_genre_sales = multi_genre_platforms.groupby(['genre',␣
 ↪'platform'])['total_sales'].sum().unstack()

# Visualización
multi_genre_sales.plot(kind='bar', stacked=True, figsize=(12, 8))
plt.title('Ventas de géneros en múltiples plataformas')
plt.xlabel('Género')
plt.ylabel('Ventas totales (en millones)')
plt.xticks(rotation=45)
plt.legend(title='Plataforma')
plt.show()
```



[16]:
```
#Paso 3.8: Distribución de juegos por género y géneros rentables

# Contar la cantidad de juegos por género
```

```
genre_count = relevant_data['genre'].value_counts()

# Calcular las ventas totales por género
genre_sales = relevant_data.groupby('genre')['total_sales'].sum().
 ↪sort_values(ascending=False)

# Visualización de la distribución de géneros y sus ventas
fig, axes = plt.subplots(1, 2, figsize=(14, 6))
genre_count.plot(kind='bar', ax=axes[0], title="Distribución de juegos por␣
 ↪género")
axes[0].set_xlabel("Género")
axes[0].set_ylabel("Cantidad de juegos")

genre_sales.plot(kind='bar', ax=axes[1], title="Ventas totales por género")
axes[1].set_xlabel("Género")
axes[1].set_ylabel("Ventas totales (en millones)")
plt.show()

#Interpretación: Identificamos géneros con ventas altas y bajas, ayudando a␣
 ↪generalizar sobre los géneros más rentables y su popularidad.
```



[17]:
```
#Paso 4: Crea un perfil de usuario para cada región
```

[18]:
```
#4.1: Las cinco plataformas principales

# Crear un DataFrame para las ventas por plataforma en cada región
```

```
platform_sales_by_region = games_df.groupby('platform')[['na_sales',␣
 ↪'eu_sales', 'jp_sales']].sum()

# Obtener las cinco plataformas principales por región
top_platforms_na = platform_sales_by_region['na_sales'].nlargest(5)
top_platforms_eu = platform_sales_by_region['eu_sales'].nlargest(5)
top_platforms_jp = platform_sales_by_region['jp_sales'].nlargest(5)

print("Top 5 plataformas en NA:\n", top_platforms_na)
print("\nTop 5 plataformas en EU:\n", top_platforms_eu)
print("\nTop 5 plataformas en JP:\n", top_platforms_jp)
```

```
Top 5 plataformas en NA:
 platform
X360    602.47
PS2     583.84
Wii     496.90
PS3     393.49
DS      382.40
Name: na_sales, dtype: float64

Top 5 plataformas en EU:
 platform
PS2     339.29
PS3     330.29
X360    270.76
Wii     262.21
PS      213.61
Name: eu_sales, dtype: float64

Top 5 plataformas en JP:
 platform
DS      175.57
PS      139.82
PS2     139.20
SNES    116.55
3DS     100.67
Name: jp_sales, dtype: float64
```

[19]:
```
#4.2: Los cinco géneros principales

# Crear un DataFrame para las ventas por género en cada región
genre_sales_by_region = games_df.groupby('genre')[['na_sales', 'eu_sales',␣
 ↪'jp_sales']].sum()

# Obtener los cinco géneros principales por región
top_genres_na = genre_sales_by_region['na_sales'].nlargest(5)
```

```
top_genres_eu = genre_sales_by_region['eu_sales'].nlargest(5)
top_genres_jp = genre_sales_by_region['jp_sales'].nlargest(5)

print("Top 5 géneros en NA:\n", top_genres_na)
print("\nTop 5 géneros en EU:\n", top_genres_eu)
print("\nTop 5 géneros en JP:\n", top_genres_jp)
```

```
Top 5 géneros en NA:
 genre
Action      879.01
Sports      684.43
Shooter     592.24
Platform    445.50
Misc        407.27
Name: na_sales, dtype: float64

Top 5 géneros en EU:
 genre
Action     519.13
Sports     376.79
Shooter    317.34
Racing     236.51
Misc       212.74
Name: eu_sales, dtype: float64

Top 5 géneros en JP:
 genre
Role-Playing    355.41
Action          161.43
Sports          135.54
Platform        130.83
Misc            108.11
Name: jp_sales, dtype: float64
```

[20]:
```
#4.3: Impacto de las clasificaciones ESRB

# Agrupar ventas por clasificación ESRB en cada región
esrb_sales = games_df.groupby('rating')[['na_sales', 'eu_sales', 'jp_sales']].
 ↪sum()

print(esrb_sales)
```

```
        na_sales  eu_sales  jp_sales
rating
AO          1.26      0.61      0.00
E        1292.99    710.25    198.11
E10+      353.32    188.52     40.20
```

```
EC          1.53      0.11      0.00
K-A         2.56      0.27      1.46
M         748.48    483.97     64.24
RP          0.00      0.08      0.00
T         759.75    427.03    151.40
```

Comentario del revisor:

Excelente! Con este análisis por región ayuda a complementar el análisis general anterior y a hacer zoom a los resultados por cada una de las regiones.

[27]:
```
#Paso 5: Prueba de hipótesis

from scipy.stats import levene

# Prueba de Levene para comparar las varianzas
levene_stat, p_value_levene = levene(xbox_scores, pc_scores)
print(f'Levene Statistic: {levene_stat}, P-value (Levene): {p_value_levene}')

# Interpretación de resultados
if p_value_levene < 0.05:
    print("Rechazamos la hipótesis nula: las varianzas no son iguales.")
    equal_var = False
else:
    print("No rechazamos la hipótesis nula: las varianzas son iguales.")
    equal_var = True
```

```
Levene Statistic: 0.07428252537501923, P-value (Levene): 0.7852257179851689
No rechazamos la hipótesis nula: las varianzas son iguales.
```

[28]:
```
#Paso 5.1

# Prueba t considerando el resultado de la prueba de Levene
t_stat, p_value = stats.ttest_ind(xbox_scores, pc_scores, equal_var=equal_var)
print(f'T-statistic (ajustada): {t_stat}, P-value (ajustada): {p_value}')

# Interpretación de resultados
if p_value < 0.05:
    print("Rechazamos la hipótesis nula: hay una diferencia significativa entre␣
 ↪las calificaciones de Xbox One y PC.")
else:
    print("No rechazamos la hipótesis nula: no hay una diferencia significativa␣
 ↪entre las calificaciones de Xbox One y PC.")
```

```
T-statistic (ajustada): -4.897976120912077, P-value (ajustada):
1.0376718408369763e-06
Rechazamos la hipótesis nula: hay una diferencia significativa entre las
calificaciones de Xbox One y PC.
```

Comentario revisor

Para estas pruebas te recomiendo hacer una prueba de Levene para mostrar si las varianzas son iguales y agregarlo dentro de la función. Actualmente lo colocar como equal_var=True. Para esto, primero debes calcular las varianzas para cada uno de las plataformas y en un segundo tiempo debes de hacer uso de la siguiente función:

```
levene(xbox_scores, pc_scores)
```

Adoptaremos un nivel de significancia de 0.05; si el valor p resultante es mayor a 0.05, no podemos rechazar la hipótesis nula, y si es menor a 0.05, rechazamos la hipótesis nula, indicando que las varianzas no son iguales.

Solamente recuerda que la prueba de levene no es sustituto a la prueba de st.ttest_ind, más bien es complemento para saber que colocar dentro del elemento "equal_var". En este caso como rechazamos la hipótesis de varianzas iguales debemos de colocar False. Es por eso que para terminar la prueba debes de realizar la prueba de st.ttest_ind considernado el resultado de la prueba realizada de levene

LISTO!

Comentario del revisor:

Jafet, excelente trabajo con la prueba de levene para mostrar el supuesto de varianzas iguales!

#Paso 6 # Conclusiones del Proyecto de Análisis de Datos de Videojuegos

Este análisis se ha enfocado en el conjunto de datos de videojuegos de la tienda online Ice, con el objetivo de identificar patrones que puedan predecir el éxito de los juegos y ayudar en la planificación de campañas publicitarias para el año 2017. A continuación, se presentan las conclusiones principales extraídas del análisis:

0.1 1. Análisis de Ventas por Plataforma

- **Principales Plataformas**: Las plataformas que han generado las mayores ventas totales incluyen **PlayStation 4**, **Xbox One** y **PC**. Esto indica que los consumidores tienden a preferir estas plataformas, lo cual debe ser considerado al planear lanzamientos futuros y campañas de marketing.
- **Caída de Plataformas**: Algunas plataformas, como **Wii** y **PS Vita**, han mostrado una disminución significativa en ventas, sugiriendo que podrían estar perdiendo relevancia en el mercado.

0.2 2. Análisis de Géneros

- **Géneros Rentables**: Los géneros más rentables han sido **Acción**, **Aventura**, y **Deportes**, indicando una fuerte preferencia de los usuarios por estos tipos de juegos.
- **Diferencias en Calificaciones**: Se encontró una diferencia significativa en las calificaciones promedio entre los géneros de **Acción** y **Deportes**, lo que sugiere que los jugadores perciben estos géneros de manera diferente, lo cual puede influir en su decisión de compra.

0.3 3. Impacto de las Calificaciones ESRB

- **Influencia en Ventas**: Las calificaciones de ESRB han demostrado tener un impacto en las ventas en diferentes regiones. Por ejemplo, los juegos con calificaciones apropiadas para la edad parecen vender mejor en todas las regiones, subrayando la importancia de la clasificación en la estrategia de marketing.

0.4 4. Comparación de Calificaciones entre Plataformas

- **Calificaciones de Usuarios**: La prueba de hipótesis reveló que las calificaciones promedio de los usuarios para las plataformas **Xbox One** y **PC** son significativamente diferentes, lo que indica que los jugadores tienen experiencias distintas dependiendo de la plataforma utilizada.

0.5 Recomendaciones

- **Estrategias de Marketing**: Dado que las plataformas como **PlayStation 4** y **Xbox One** están en auge, se recomienda centrar las campañas de marketing en juegos desarrollados para estas plataformas.
- **Desarrollo de Nuevos Títulos**: Se sugiere investigar y desarrollar juegos en géneros populares como **Acción** y **Deportes**, que han mostrado un alto potencial de ventas.
- **Monitoreo de Tendencias**: Es importante implementar un sistema para el análisis continuo de los datos de ventas y tendencias del mercado, permitiendo ajustes en tiempo real a las estrategias de marketing y desarrollo de productos.

En conclusión, este análisis proporciona un conjunto valioso de información que puede ser utilizada para mejorar la toma de decisiones en la tienda online Ice, ayudando a identificar oportunidades de crecimiento y desarrollo en el competitivo mercado de videojuegos.

Comentario revisor

Jafet, en general creo que hiciste un muy buen trabajo con el proyecto, pudiste limpiar y trabajar las bases de datos de beuna manera. Además, el análisis explorario de datos fue completo al mostrar resultados relevantes que pueden ser de mucha utilidad para la toma de decisiones y desarrollaste las pruebas de hipótesis de una buena manera. No obstante, recuerda que siempre podemos mejorar y te menciono algunos puntos que debes considerar:

- Verificar que cuando llenamos variables con valores nulos los estamos comletando con valores que no sesgan nuestros resultados
- Considerar eliminar registros atipicos que puedan sesgar nuestros resultados.
- Verificar los supuestos de las pruebas de hipótesis.