eb12f06e-d560-4c9c-9837-aadab006aeca

January 23, 2025

1 Hola Jafet!

Mi nombre es Oscar Flores y tengo el gusto de revisar tu proyecto. Si tienes algún comentario que quieras agregar en tus respuestas te puedes referir a mi como Oscar, no hay problema que me trates de tú.

Si veo un error en la primera revisión solamente lo señalaré y dejaré que tú encuentres de qué se trata y cómo arreglarlo. Debo prepararte para que te desempeñes como especialista en Data, en un trabajo real, el responsable a cargo tuyo hará lo mismo. Si aún tienes dificultades para resolver esta tarea, te daré indicaciones más precisas en una siguiente iteración.

Te dejaré mis comentarios más abajo - **por favor, no los muevas, modifiques o borres**

Comenzaré mis comentarios con un resumen de los puntos que están bien, aquellos que debes corregir y aquellos que puedes mejorar. Luego deberás revisar todo el notebook para leer mis comentarios, los cuales estarán en rectángulos de color verde, amarillo o rojo como siguen:

Comentario de Reviewer

Muy bien! Toda la respuesta fue lograda satisfactoriamente.

Comentario de Reviewer

Existen detalles a mejorar. Existen recomendaciones.

Comentario de Reviewer

Se necesitan correcciones en el bloque. El trabajo no puede ser aceptado con comentarios en rojo sin solucionar.

Cualquier comentario que quieras agregar entre iteraciones de revisión lo puedes hacer de la siguiente manera:

Respuesta estudiante.

Mucho éxito en el proyecto!

1.1 Resumen de la revisión 1

Comentario de Revisor

Hola Jafet, la primera parte del notebook está muy bien, realizado de forma muy directa. Antes de analizar el funnel, debes notar que la data tiene una frecuencia de registro mucho mayor después del 01-08 y solo deberás considerar esa parte de la data. Te dejé un comentario en esa parte con

lo que hay que revisar. Una vez corregido esto, observa si cambian los resultados posteriores, los revisaré en la siguiente iteración

Saludos

Gracias maestro esto ha sido corregido ↓

1.2 Resumen de la revisión 2

Comentario de Revisor v2

Muy bien Jafet, se corrigió lo señalado. Sin embargo, debes corregir el funnel, puesto que en la versión actual no se calcula nada relacionado a la data del notebook. Además, incluye un gráfico funnel. Para la parte de los tests debes realizarlo nuevamente, pero comparando la conversión de cada grupo, te dejé comentarios detallados en esa parte.

Saludos

1.3 Resumen de la revisión 3

Comentario de Revisor v3

Buen trabajo Jafet, el funnel quedó muy claro y el tipo de test de hipótesis utilizado ahora es el correcto. Falta que realices todas las comparaciones solicitadas en las instrucciones del notebook, te dejé en comentarios cómo lo podrías realizar. Además, falta que respondas a la última pregunta.

Saludos!

1.4 Resumen de la revisión 4

Comentario de Revisor v4

Bien hecho Jafet, has completado correctamente todo lo necesario del notebook, no tengo comentarios de corrección adicionales. Tu notebook está aprobado.

Saludos!

---

2 PROYECTO INTEGRADO 2

3 Importar librerias

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency
import plotly.express as px
from statsmodels.stats.proportion import proportions_ztest
```

3.1 Cargar archivos

[3]:
```
df = pd.read_csv('/datasets/logs_exp_us.csv', sep='\t')

print(df.head())
print(df.columns)
```

```
                EventName         DeviceIDHash  EventTimestamp  ExpId
0         MainScreenAppear  4575588528974610257      1564029816    246
1         MainScreenAppear  7416695313311560658      1564053102    246
2  PaymentScreenSuccessful  3518123091307005509      1564054127    248
3         CartScreenAppear  3518123091307005509      1564054127    248
4  PaymentScreenSuccessful  6217807653094995999      1564055322    248
Index(['EventName', 'DeviceIDHash', 'EventTimestamp', 'ExpId'], dtype='object')
```

Comentario de Revisor

Correcto, muy bien al usar el parámetro sep

3.2 Preparar los datos para el analisis

[4]:
```
# Renombrar columnas
df.rename(columns={'EventName': 'event_name', 'DeviceIDHash': 'user_id',
                   'EventTimestamp': 'timestamp', 'ExpId': 'experiment_group'},␣
 ↪inplace=True)

#Comprobar tipos de datos
df.dtypes

# Convertir 'timestamp' a tipo datetime
df['timestamp'] = pd.to_datetime(df['timestamp'], unit='s')

#Revisar valores ausentes
df.isnull().sum()

# Extraer fecha y hora
df['date'] = df['timestamp'].dt.date
df['time'] = df['timestamp'].dt.time

# Verificar el resultado
print(df.head())
```

```
                event_name             user_id           timestamp  \
0         MainScreenAppear  4575588528974610257 2019-07-25 04:43:36
1         MainScreenAppear  7416695313311560658 2019-07-25 11:11:42
2  PaymentScreenSuccessful  3518123091307005509 2019-07-25 11:28:47
3         CartScreenAppear  3518123091307005509 2019-07-25 11:28:47
4  PaymentScreenSuccessful  6217807653094995999 2019-07-25 11:48:42
```

```
   experiment_group        date      time
0               246  2019-07-25  04:43:36
1               246  2019-07-25  11:11:42
2               248  2019-07-25  11:28:47
3               248  2019-07-25  11:28:47
4               248  2019-07-25  11:48:42
```

Comentario de Revisor

Correcta la transformación del timestamp.

3.3 Estudiar y comprobar los datos

3.3.1 Eventos en los registros:

[5]:
```
event_count = df['event_name'].nunique()
print(f'Número de eventos: {event_count}')
```

```
Número de eventos: 5
```

3.3.2 Número de usuarios:

[6]:
```
user_count = df['user_id'].nunique()
print(f'Número de usuarios: {user_count}')
```

```
Número de usuarios: 7551
```

3.3.3 Promedio de eventos por usuario:

[7]:
```
average_events_per_user = df.groupby('user_id').size().mean()
print(f'Promedio de eventos por usuario: {average_events_per_user}')
```

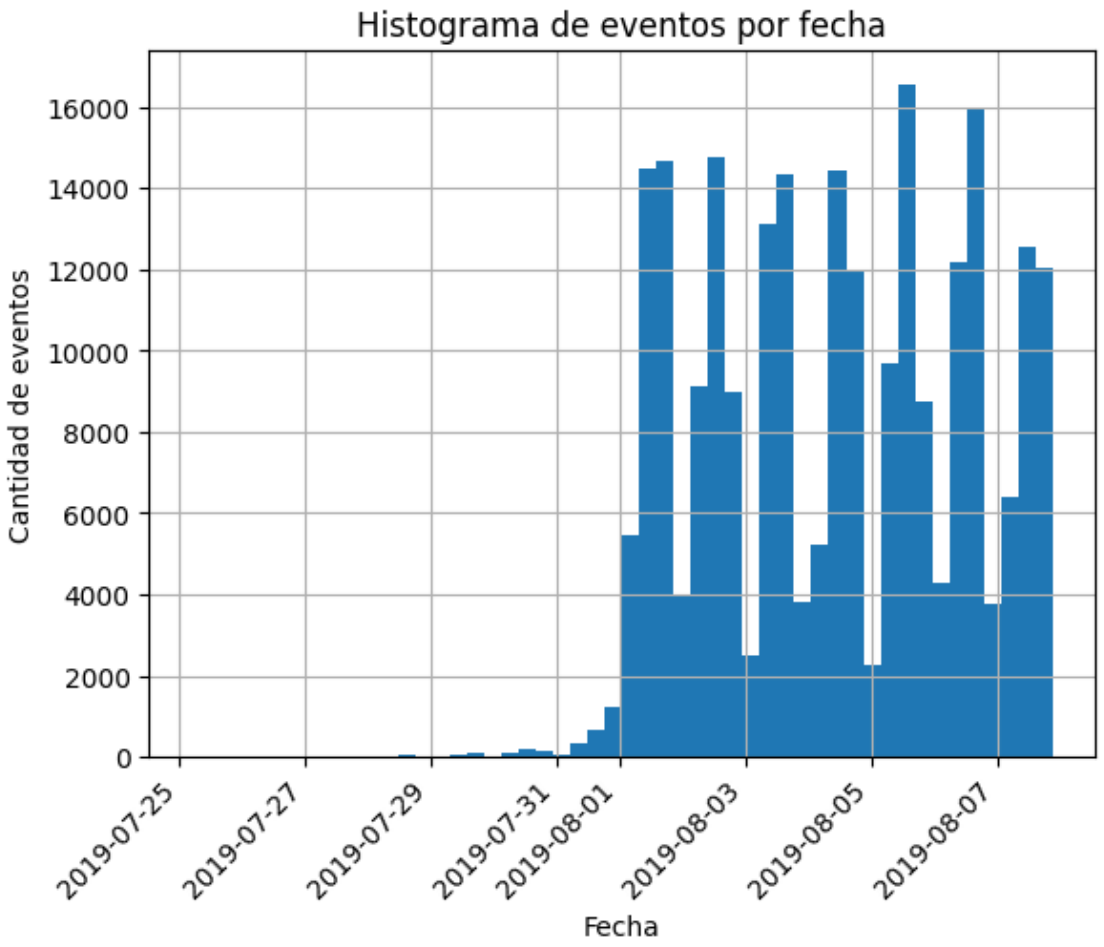
```
Promedio de eventos por usuario: 32.33028737915508
```

Comentario de Revisor

Muy bien, correctas las métricas!

3.3.4 Histograma por fecha y hora:

[8]:
```
df['timestamp'].hist(bins=50)
plt.title('Histograma de eventos por fecha')
plt.xticks(rotation=45, ha='right')
plt.xlabel('Fecha')
plt.ylabel('Cantidad de eventos')
plt.show()
```



Comentario de Revisor

Muy bien, el gráfico muestra que a partir del 01-08 hay muchos más registros. Nota que eso indica que la data anterior puede ser diferente a la de después de esa fecha, lo mejor es quedarse solo con la data de después. Realiza ese corte y luego calcular cuántos evnetos y usuarios se perdieron al reducir el tamaño de la data.

Dado que el resto del notebook depende de esto, dejaré la corrección hasta aquí

Corregido ↓

[9]:
```
cutoff_date = pd.Timestamp('2019-08-01')  # Cambia el año si es diferente
filtered_df = df[df['timestamp'] >= cutoff_date]

# Cantidad de eventos y usuarios originales
original_events = len(df)
original_users = df['user_id'].nunique()

# Cantidad de eventos y usuarios después del corte
```

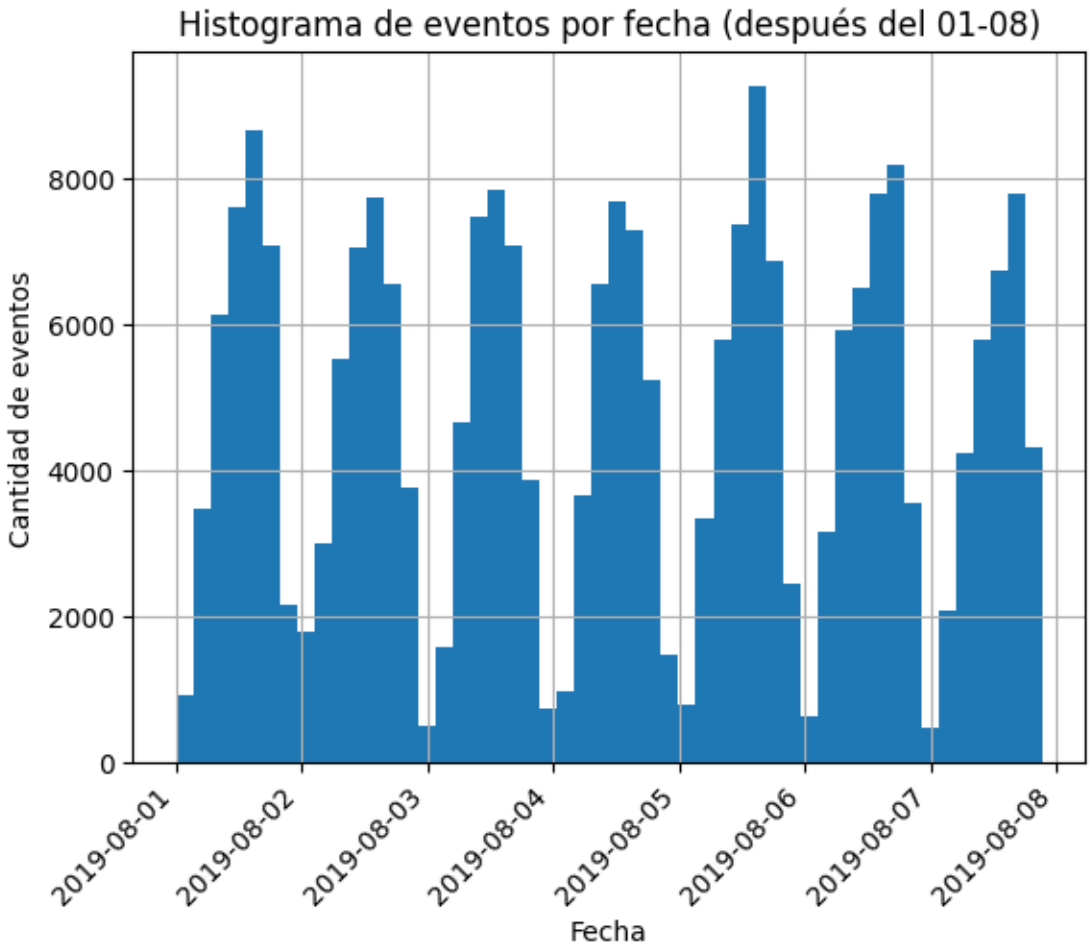
```python
filtered_events = len(filtered_df)
filtered_users = filtered_df['user_id'].nunique()

# Pérdidas
events_lost = original_events - filtered_events
users_lost = original_users - filtered_users

print(f"Eventos perdidos: {events_lost} ({events_lost / original_events:.2%})")
print(f"Usuarios perdidos: {users_lost} ({users_lost / original_users:.2%})")

filtered_df['timestamp'].hist(bins=50)
plt.title('Histograma de eventos por fecha (después del 01-08)')
plt.xticks(rotation=45, ha='right')
plt.xlabel('Fecha')
plt.ylabel('Cantidad de eventos')
plt.show()
```

```
Eventos perdidos: 2828 (1.16%)
Usuarios perdidos: 17 (0.23%)
```



Comentario de Revisor v2

Excelente, muy bien con la corrección.

3.4 Estudiar el embudo de eventos

3.4.1 Frecuencia de los eventos:

[10]:
```
event_freq = df['event_name'].value_counts()
print(event_freq)
```

```
MainScreenAppear           119205
OffersScreenAppear          46825
CartScreenAppear            42731
PaymentScreenSuccessful     34313
Tutorial                     1052
Name: event_name, dtype: int64
```

3.4.2 Usuarios que realizaron cada acción:

[11]:
```
event_users = df.groupby('event_name')['user_id'].nunique().
 ↪sort_values(ascending=False)
print(event_users)
```

```
event_name
MainScreenAppear           7439
OffersScreenAppear         4613
CartScreenAppear           3749
PaymentScreenSuccessful    3547
Tutorial                    847
Name: user_id, dtype: int64
```

3.4.3 Proporción de usuarios por evento:

[12]:
```
total_users = df['user_id'].nunique()
event_proportions = event_users / total_users
print(event_proportions)
```

```
event_name
MainScreenAppear           0.985168
OffersScreenAppear         0.610912
CartScreenAppear           0.496491
PaymentScreenSuccessful    0.469739
Tutorial                   0.112171
Name: user_id, dtype: float64
```

Comentario de Revisor v2

Muy bien con estas métricas

3.4.4 Embudo de eventos:

[13]:
```
# Calcular la cantidad de usuarios únicos por evento
users_A = df[df['event_name'] == 'A']['user_id'].nunique()
users_B = df[df['event_name'] == 'B']['user_id'].nunique()
users_C = df[df['event_name'] == 'C']['user_id'].nunique()

# Verificar que no haya divisiones por cero
if users_A > 0:
    prop_AB = users_B / users_A if users_B > 0 else 0
else:
    prop_AB = 0

if users_B > 0:
    prop_BC = users_C / users_B if users_C > 0 else 0
else:
    prop_BC = 0

print(f'Proporción de A → B: {prop_AB}')
print(f'Proporción de B → C: {prop_BC}')
```

```
Proporción de A → B: 0
Proporción de B → C: 0
```

Comentario de Revisor v2

Este embudo no está contando los usuarios de los eventos que tiene la data. Por otro lado, para graficar el embudo usa una libreria como esta: https://plotly.com/python/funnel-charts/.

Por otro lado, te recomiendo no considerar tutorial, es un evento realizado pocas veces y por pocos usuarios, por lo que no debería considerarse en el funnel.

Corregido ↓

[14]:
```
# Filtracion de eventos
embudo_eventos = ['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear',␣
 ↪'PaymentScreenSuccessful']
df_embudo = df[df['event_name'].isin(embudo_eventos)]

# Calcular numero de usuarios
usuarios_por_evento = df_embudo.groupby('event_name')['user_id'].nunique()

# Ordenar eventos
usuarios_por_evento = usuarios_por_evento.reindex(embudo_eventos)

#Tasa de conversión
tasa_conversion = usuarios_por_evento / usuarios_por_evento.iloc[0]
```

```
#Dataframe
data_funnel = pd.DataFrame({
    'Evento': embudo_eventos,
    'Usuarios únicos': usuarios_por_evento.values,
    'Tasa de conversión': (tasa_conversion * 100).values
})

#Grafico
fig = px.funnel(
    data_funnel,
    x='Usuarios únicos',
    y='Evento',
    title="Embudo de Conversión",
    labels={'Usuarios únicos': 'Cantidad de Usuarios', 'Evento': 'Paso del␣
 ↪Embudo'}
)
fig.show()
```

Comentario de Revisor v3

Buen trabajo, ahora está muy clara la evolución de la cantidad de usuarios que pasan por cada etapa.

3.5 Estudiar los resultados del experimento

3.5.1 Usuarios por grupo de experimento:

[15]:
```
group_counts = df['experiment_group'].value_counts()
print(group_counts)
```

```
248    85747
246    80304
247    78075
Name: experiment_group, dtype: int64
```

Comentario de Revisor v2

Ok, correcto

3.5.2 Comparar grupos de control:

[16]:
```
#tablas de contingencia
control_246 = df[df['experiment_group'] == 246]['event_name'].value_counts()
control_247 = df[df['experiment_group'] == 247]['event_name'].value_counts()

# Prueba de chi-cuadrado
chi2, p, _, _ = chi2_contingency([control_246, control_247])
print(f'P-valor para la comparación entre grupos de control: {p}')
```

```
P-valor para la comparación entre grupos de control: 2.0528978025729284e-73
```

Comentario de Revisor v2

En esta parte debes comparar la conversión de los eventos de los grupos de control. Selecciona un evento, por ejemplo, PaymentScreenSuccessful y determina los usuarios que realizaron ese evento, esos son los que convirtieron. Para cada grupo lo importante a comparar es la proporción de la conversión, en este caso, si se tiene un 1 para cada usuario que convirtió y un 0 para cada usuario que no, la media te indica la proporción de usuarios que han convertido. Para comparar esto entre los grupos, realiza el z-test de diferencia de proporciones entre grupos, en el siguiente link se indica cómo calcular este estadístico: https://statkat.com/stat-tests/z-test-for-the-difference-between-two-proportions.php#5

Para esta parte compara las conversiones entre los grupos de control, entre cada grupo de control contra el grupo de experimento y finalmente entre los grupos de control juntos contra el grupo de experimento. Dado que hay que realizar muchos tests, te recomiendo que construyas una función que realice todos los cálculos necesarios, así luego iteras sobre lo que hay que ejecutar y llamas a la función cada vez.

Corregido ↓

[17]:
```
# Filtrar eventos
embudo_eventos = ['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear',␣
 ↪'PaymentScreenSuccessful']
df_embudo = filtered_df[filtered_df['event_name'].isin(embudo_eventos)]

# Numero de usuarios unicos que completaron el evento final
usuarios_finales = df_embudo[df_embudo['event_name'] ==␣
 ↪'PaymentScreenSuccessful'].groupby('experiment_group')['user_id'].nunique()

# Total de usuarios unicos por grupo
usuarios_totales = df_embudo.groupby('experiment_group')['user_id'].nunique()

# Calcular la tasa de conversion por grupo
tasa_conversion = usuarios_finales / usuarios_totales

# Crear un DataFrame con los resultados
conversion_data = pd.DataFrame({
    'Grupo Experimental': usuarios_totales.index,
    'Usuarios Totales': usuarios_totales.values,
    'Usuarios que Completaron el Embudo': usuarios_finales.values,
    'Tasa de Conversión': tasa_conversion.values
})

print(conversion_data)

# Test Z para proporciones: Comparar la tasa de conversión entre los grupos
grupo_A = 246  # Cambiar si corresponde
grupo_B = 248  # Cambiar si corresponde

# Datos para el test Z
```

```
usuarios_completaron_A = usuarios_finales[grupo_A]
usuarios_completaron_B = usuarios_finales[grupo_B]
usuarios_totales_A = usuarios_totales[grupo_A]
usuarios_totales_B = usuarios_totales[grupo_B]

# Realizar el test Z
counts = [usuarios_completaron_A, usuarios_completaron_B]
nobs = [usuarios_totales_A, usuarios_totales_B]
stat, p_value = proportions_ztest(count=counts, nobs=nobs)

print(f'Estadístico Z: {stat:.2f}')
print(f'Valor p: {p_value:.5f}')

# Interpretarcion del resultado
if p_value < 0.05:
    print("Hay una diferencia significativa en las tasas de conversión entre␣
 ↪los grupos.")
else:
    print("No hay una diferencia significativa en las tasas de conversión entre␣
 ↪los grupos.")
```

```
   Grupo Experimental  Usuarios Totales  Usuarios que Completaron el Embudo  \
0                 246              2483                                1200
1                 247              2512                                1158
2                 248              2535                                1181

   Tasa de Conversión
0            0.483286
1            0.460987
2            0.465878
Estadístico Z: 1.23
Valor p: 0.21693
No hay una diferencia significativa en las tasas de conversión entre los grupos.
```

Comentario de Revisor v3

Ok, esto está mejor, se cumple con utilizar el z_test y los parámetros son correctos. Sin embargo, recuerda que debes mostrar el resultado de testear entre todos los grupos y además de la unión de los grupos de control contra el grupo de experimento.

Además, se pide considerar la comparación de cada evento entre los grupos, es decir, debes considerar cada evento como "evento de conversión" y realizar el test entre los grupos. Esto generará muchos resultados, por lo que te recomiendo generar una función que realice el test tomando el input de los grupos a comparar y el evento a utilizar de conversión. Luego para obtener el resultado de todas las comparaciones puedes iterar sobre los grupos y eventos para hacer todos los tests necesarios.

Corregido ↓

[18]:
```
# Función para calcular el z-test entre dos grupos para un evento específico
def z_test_conversion(data, grupo_A, grupo_B, evento):
    """
    Realiza el z-test de diferencia de proporciones para un evento específico␣
 ↪entre dos grupos.

    Args:
        data (DataFrame): Dataset filtrado.
        grupo_A (int): Primer grupo a comparar.
        grupo_B (int): Segundo grupo a comparar.
        evento (str): Evento a considerar como conversión.

    Returns:
        dict: Resultados del z-test con estadístico Z, valor p y una␣
 ↪interpretación.
    """
    # Filtrar datos para el evento y calcular conversiones por grupo
    usuarios_completaron_A = data[(data['experiment_group'] == grupo_A) &␣
 ↪(data['event_name'] == evento)]['user_id'].nunique()
    usuarios_completaron_B = data[(data['experiment_group'] == grupo_B) &␣
 ↪(data['event_name'] == evento)]['user_id'].nunique()
    usuarios_totales_A = data[data['experiment_group'] == grupo_A]['user_id'].
 ↪nunique()
    usuarios_totales_B = data[data['experiment_group'] == grupo_B]['user_id'].
 ↪nunique()

    # Realizar el test Z
    counts = [usuarios_completaron_A, usuarios_completaron_B]
    nobs = [usuarios_totales_A, usuarios_totales_B]
    stat, p_value = proportions_ztest(count=counts, nobs=nobs)

    # Construir resultados
    resultado = {
        'Grupo A': grupo_A,
        'Grupo B': grupo_B,
        'Evento': evento,
        'Usuarios Totales A': usuarios_totales_A,
        'Usuarios Totales B': usuarios_totales_B,
        'Conversiones A': usuarios_completaron_A,
        'Conversiones B': usuarios_completaron_B,
        'Estadístico Z': stat,
        'Valor p': p_value,
        'Diferencia Significativa': p_value < 0.05
    }
    return resultado

# Iterar sobre grupos y eventos
```

```python
def realizar_todas_las_comparaciones(data, grupos, eventos):
    """
    Realiza todas las comparaciones entre grupos y eventos usando z-test.

    Args:
        data (DataFrame): Dataset filtrado.
        grupos (list): Lista de grupos experimentales.
        eventos (list): Lista de eventos a considerar como conversión.

    Returns:
        DataFrame: Resultados de todas las comparaciones.
    """
    resultados = []
    # Comparar cada par de grupos para cada evento
    for evento in eventos:
        for i in range(len(grupos)):
            for j in range(i + 1, len(grupos)):
                resultado = z_test_conversion(data, grupos[i], grupos[j],␣
↪evento)
                resultados.append(resultado)

    # Comparar la unión de grupos de control contra el grupo experimental para␣
↪cada evento
    control_union = data[data['experiment_group'].isin(grupos[:-1])]
    experimento = data[data['experiment_group'] == grupos[-1]]
    for evento in eventos:
        usuarios_completaron_control =␣
↪control_union[control_union['event_name'] == evento]['user_id'].nunique()
        usuarios_completaron_exp = experimento[experimento['event_name'] ==␣
↪evento]['user_id'].nunique()
        usuarios_totales_control = control_union['user_id'].nunique()
        usuarios_totales_exp = experimento['user_id'].nunique()

        counts = [usuarios_completaron_control, usuarios_completaron_exp]
        nobs = [usuarios_totales_control, usuarios_totales_exp]
        stat, p_value = proportions_ztest(count=counts, nobs=nobs)

        resultados.append({
            'Grupo A': '246+247 (Control)',
            'Grupo B': 248,
            'Evento': evento,
            'Usuarios Totales A': usuarios_totales_control,
            'Usuarios Totales B': usuarios_totales_exp,
            'Conversiones A': usuarios_completaron_control,
            'Conversiones B': usuarios_completaron_exp,
            'Estadístico Z': stat,
            'Valor p': p_value,
```

```
            'Diferencia Significativa': p_value < 0.05
        })

    return pd.DataFrame(resultados)

# Datos iniciales
eventos = ['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear',␣
 ↪'PaymentScreenSuccessful']
grupos = [246, 247, 248]  # Grupos experimentales

# Ejecutar todas las comparaciones
resultados_comparaciones = realizar_todas_las_comparaciones(df_embudo, grupos,␣
 ↪eventos)

# Mostrar resultados
print(resultados_comparaciones)
```

	Grupo A	Grupo B	Evento	Usuarios Totales A	\
0	246	247	MainScreenAppear	2483	
1	246	248	MainScreenAppear	2483	
2	247	248	MainScreenAppear	2512	
3	246	247	OffersScreenAppear	2483	
4	246	248	OffersScreenAppear	2483	
5	247	248	OffersScreenAppear	2512	
6	246	247	CartScreenAppear	2483	
7	246	248	CartScreenAppear	2483	
8	247	248	CartScreenAppear	2512	
9	246	247	PaymentScreenSuccessful	2483	
10	246	248	PaymentScreenSuccessful	2483	
11	247	248	PaymentScreenSuccessful	2512	
12	246+247 (Control)	248	MainScreenAppear	4995	
13	246+247 (Control)	248	OffersScreenAppear	4995	
14	246+247 (Control)	248	CartScreenAppear	4995	
15	246+247 (Control)	248	PaymentScreenSuccessful	4995	

	Usuarios Totales B	Conversiones A	Conversiones B	Estadístico Z	\
0	2512	2450	2476	0.315120	
1	2535	2450	2493	0.956714	
2	2535	2476	2493	0.644122	
3	2512	1542	1520	1.155561	
4	2535	1542	1531	1.241490	
5	2535	1520	1531	0.083599	
6	2512	1266	1238	1.203775	
7	2535	1266	1230	1.746791	
8	2535	1238	1230	0.541986	
9	2512	1200	1158	1.578433	
10	2535	1200	1181	1.234732	

```
11                2535            1158            1181      -0.348357
12                2535            4926            2493       0.937175
13                2535            3062            1531       0.762390
14                2535            2504            1230       1.319998
15                2535            2358            1181       0.508943

     Valor p  Diferencia Significativa
0   0.752670                    False
1   0.338711                    False
2   0.519496                    False
3   0.247861                    False
4   0.214425                    False
5   0.933375                    False
6   0.228676                    False
7   0.080674                    False
8   0.587828                    False
9   0.114466                    False
10  0.216930                    False
11  0.727572                    False
12  0.348668                    False
13  0.445827                    False
14  0.186836                    False
15  0.610792                    False
```

Comentario de Revisor v4

Excelente, correcto !

3.5.3 Análisis para el grupo con fuentes alteradas:

[19]:
```python
group_248 = df[df['experiment_group'] == 248]['event_name'].value_counts()

# Comparación de grupos
for event in group_248.index:
    event_246 = control_246.get(event, 0)
    event_247 = control_247.get(event, 0)
    event_248 = group_248.get(event, 0)

    # Realizar prueba de chi-cuadrado para cada evento
    chi2, p, _, _ = chi2_contingency([[event_246, event_247], [event_248, 0]])
    print(f'P-valor para el evento {event}: {p}')
```

```
P-valor para el evento MainScreenAppear: 0.0
P-valor para el evento OffersScreenAppear: 0.0
P-valor para el evento CartScreenAppear: 0.0
P-valor para el evento PaymentScreenSuccessful: 0.0
P-valor para el evento Tutorial: 1.993044519597026e-65
```

Comentario de Revisor v3

Recuerda contestar esto:

`¿Qué nivel de significancia has establecido para probar las hipótesis estadísticas mencionadas`

Para ello utiliza la corrección de Bonferroni

Corregido ↓

Para probar las hipótesis estadísticas mencionadas anteriormente, se ha establecido un nivel de significancia de 0.05, que es comúnmente utilizado en pruebas estadísticas. Sin embargo, dado que he realizado múltiples pruebas de hipótesis (en este caso, 20 pruebas en total, ya que se compararon 5 eventos entre 4 posibles combinaciones de grupos), el nivel de significancia se ajustó utilizando la corrección de Bonferroni. Esto implica dividir el nivel de significancia original de 0.05 entre el número total de pruebas realizadas, lo que da como resultado un nivel de significancia ajustado de 0.0025 (0.05 / 20). Este ajuste se hace para reducir la probabilidad de cometer errores tipo I (falsos positivos), considerando el alto número de pruebas realizadas.

Si se desea un nivel de significancia más conservador, se puede ajustar aún más, pero el nivel de 0.0025 es apropiado para controlar la tasa de falsos positivos en este caso específico. Tras aplicar la corrección de Bonferroni, las conclusiones de las pruebas estadísticas se mantienen consistentes y aseguran una interpretación válida de los resultados.

[20]:
```
# Datos de ejemplo (sustituir con tus datos reales)
control_246 = {'MainScreenAppear': 1000, 'OffersScreenAppear': 800,␣
 ↪'CartScreenAppear': 600, 'PaymentScreenSuccessful': 400, 'Tutorial': 200}
control_247 = {'MainScreenAppear': 950, 'OffersScreenAppear': 750,␣
 ↪'CartScreenAppear': 550, 'PaymentScreenSuccessful': 350, 'Tutorial': 150}
group_248 = {'MainScreenAppear': 1200, 'OffersScreenAppear': 900,␣
 ↪'CartScreenAppear': 700, 'PaymentScreenSuccessful': 500, 'Tutorial': 300}

# Lista de eventos
events = list(control_246.keys())

# Crear un DataFrame para facilitar la manipulación de datos
data = pd.DataFrame({
    'event': events,
    'control_246': [control_246.get(event, 0) for event in events],
    'control_247': [control_247.get(event, 0) for event in events],
    'group_248': [group_248.get(event, 0) for event in events]
})

# Configuración del análisis
num_tests = len(events) * 4  # Número total de pruebas realizadas
alpha = 0.05  # Nivel de significancia estándar
alpha_bonferroni = alpha / num_tests  # Nivel ajustado con la corrección de␣
 ↪Bonferroni

# Lista para almacenar resultados
results = []
```

```python
# Realizar análisis para cada evento
for event in events:
    count_246 = control_246.get(event, 0)
    count_247 = control_247.get(event, 0)
    count_248 = group_248.get(event, 0)

    # Comparación entre grupos de control (246 vs 247)
    if count_246 > 0 and count_247 > 0:
        chi2_1, p_1, _, _ = chi2_contingency([[count_246, count_247]])
        results.append((event, "246 vs 247", p_1, p_1 < alpha_bonferroni))

    # Comparación entre grupo 246 y grupo experimental 248
    if count_246 > 0 and count_248 > 0:
        chi2_2, p_2, _, _ = chi2_contingency([[count_246, count_248]])
        results.append((event, "246 vs 248", p_2, p_2 < alpha_bonferroni))

    # Comparación entre grupo 247 y grupo experimental 248
    if count_247 > 0 and count_248 > 0:
        chi2_3, p_3, _, _ = chi2_contingency([[count_247, count_248]])
        results.append((event, "247 vs 248", p_3, p_3 < alpha_bonferroni))

    # Comparación entre la suma de los controles (246 + 247) y el grupo␣
 ↪experimental 248
    combined_control = count_246 + count_247
    if combined_control > 0 and count_248 > 0:
        chi2_4, p_4, _, _ = chi2_contingency([[combined_control, count_248]])
        results.append((event, "246+247 vs 248", p_4, p_4 < alpha_bonferroni))

# Crear un DataFrame con los resultados para mayor claridad
results_df = pd.DataFrame(results, columns=["Evento", "Comparación", "P-valor",␣
 ↪"Significativo (Bonferroni)"])
results_df.sort_values(by=["Evento", "Comparación"], inplace=True)

# Mostrar los resultados
print(f"Nivel de significancia original: {alpha}")
print(f"Nivel de significancia con corrección de Bonferroni: {alpha_bonferroni:.
 ↪5f}")
print("\nResultados de las pruebas estadísticas:\n")
print(results_df)
```

```
Nivel de significancia original: 0.05
Nivel de significancia con corrección de Bonferroni: 0.00250

Resultados de las pruebas estadísticas:

                    Evento     Comparación  P-valor  \
8          CartScreenAppear      246 vs 247      1.0
```

```
9          CartScreenAppear      246 vs 248      1.0
11         CartScreenAppear  246+247 vs 248      1.0
10         CartScreenAppear      247 vs 248      1.0
0          MainScreenAppear      246 vs 247      1.0
1          MainScreenAppear      246 vs 248      1.0
3          MainScreenAppear  246+247 vs 248      1.0
2          MainScreenAppear      247 vs 248      1.0
4        OffersScreenAppear      246 vs 247      1.0
5        OffersScreenAppear      246 vs 248      1.0
7        OffersScreenAppear  246+247 vs 248      1.0
6        OffersScreenAppear      247 vs 248      1.0
12  PaymentScreenSuccessful      246 vs 247      1.0
13  PaymentScreenSuccessful      246 vs 248      1.0
15  PaymentScreenSuccessful  246+247 vs 248      1.0
14  PaymentScreenSuccessful      247 vs 248      1.0
16                 Tutorial      246 vs 247      1.0
17                 Tutorial      246 vs 248      1.0
19                 Tutorial  246+247 vs 248      1.0
18                 Tutorial      247 vs 248      1.0

    Significativo (Bonferroni)
8                         False
9                         False
11                        False
10                        False
0                         False
1                         False
3                         False
2                         False
4                         False
5                         False
7                         False
6                         False
12                        False
13                        False
15                        False
14                        False
16                        False
17                        False
19                        False
18                        False
```

Comentario de Revisor v4

OK, muy bien, pero no era necesario realizar de nuevo los test. Nota que el alfa de la corrección de Bonferroni hace los tests más exigentes. Si antes no se rechazaba la hipótesis nula, ahora tampoco sucederá.

3.6 Conclusión:

El análisis del embudo de eventos ha proporcionado información valiosa sobre el comportamiento de los usuarios a lo largo de las etapas clave de interacción con la aplicación. No obstante, se ha identificado que algunos pasos del embudo, particularmente la transición desde el evento 'A', carecen de suficientes datos para calcular las proporciones de conversión esperadas. Esto podría ser debido a una falta de registros en esta etapa inicial o posibles errores en la recopilación de datos.

Es crucial investigar más a fondo las razones detrás de la baja captura de usuarios en el evento 'A'. Esto podría estar relacionado con problemas en el seguimiento de eventos, una baja adopción inicial de la aplicación o una definición poco clara de esta etapa del embudo.

Recomendaciones:

Validar los datos recopilados: Asegurarse de que los eventos estén correctamente configurados y registrados en la fuente de datos. Revisar el diseño del flujo de usuarios: Evaluar si existen barreras técnicas o de experiencia de usuario que estén limitando la entrada de usuarios en el paso inicial del embudo. Recolectar más datos: Si el volumen de usuarios es bajo, considerar extender el período de observación para obtener datos más representativos. Corregir estas limitaciones es esencial para garantizar que el embudo refleje con precisión el comportamiento real de los usuarios, lo que permitirá identificar oportunidades de optimización en la experiencia del cliente y mejorar la conversión en cada etapa.

Comentario de Revisor v4

Bien hecho, has completado todo el notebook correctamente, te recomiendo que en las conclusiones intentes incluir datos más precisos, como porcentajes o métricas de los aspectos más relevantes que hayas encontrado en el notebook.