Sprint 11 - Proyecto

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from plotly import graph_objects as go
from scipy import stats as st
import math as mth
from scipy.stats import ttest_ind
```

Abrir el archivo de datos y leer la información general

```
logs = pd.read_csv(
    './logs_exp_us.csv',
    engine='python',
    sep='\s',
    header=0,
    dtype={'EventName':'category', 'ExpId':'category'},
)
```

```
<>:4: SyntaxWarning: invalid escape sequence '\s'
<>:4: SyntaxWarning: invalid escape sequence '\s'
/var/folders/yd/jr1h18jn6px95w7s343fr20h0000gp/T/ipykernel_4585/2198974406.py:4: Synt
axWarning: invalid escape sequence '\s'
  sep='\s',
```

- **EventName** y **ExpId** tienen muchos valores repetidos (categorías en lugar de valores únicos), la conversión a category puede reducir significativamente el uso de memoria.
- Como estoy trabajando con un dataset grande, **memory_usage='deep'** Me ayuda a optimizar mejor el uso de memoria.

```
display(logs.head())
logs.info(memory_usage='deep')
```

	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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244126 entries, 0 to 244125
Data columns (total 4 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   EventName       244126 non-null  category
 1   DeviceIDHash    244126 non-null  int64
 2   EventTimestamp  244126 non-null  int64
 3   ExpId           244126 non-null  category
dtypes: category(2), int64(2)
memory usage: 4.2 MB
```

Preparar los datos para el análisis

Columnas

```
# Modificar el nombre de las columnas de forma conveniente
logs.columns = ['event_name', 'user_id','full_date','group']
logs.head(2)
```

	event_name	user_id	full_date	group
0	MainScreenAppear	4575588528974610257	1564029816	246
1	MainScreenAppear	7416695313311560658	1564053102	246

```
logs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244126 entries, 0 to 244125
Data columns (total 4 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   event_name  244126 non-null  category
 1   user_id     244126 non-null  int64
 2   full_date   244126 non-null  int64
 3   group       244126 non-null  category
dtypes: category(2), int64(2)
memory usage: 4.2 MB
```

Duplicados

```
# Verificamos si hay duplicados explicitos
logs.duplicated().sum()
```

```
np.int64(413)
```

```
logs[logs.duplicated()]
```

	event_name	user_id	full_date	group
453	MainScreenAppear	5613408041324010552	1564474784	248
2350	CartScreenAppear	1694940645335807244	1564609899	248
3573	MainScreenAppear	434103746454591587	1564628377	248
4076	MainScreenAppear	3761373764179762633	1564631266	247
4803	MainScreenAppear	2835328739789306622	1564634641	248
...	...	...	...	...
242329	MainScreenAppear	8870358373313968633	1565206004	247
242332	PaymentScreenSuccessful	4718002964983105693	1565206005	247
242360	PaymentScreenSuccessful	2382591782303281935	1565206049	246
242362	CartScreenAppear	2382591782303281935	1565206049	246
242635	MainScreenAppear	4097782667445790512	1565206618	246

413 rows × 4 columns

- **Como el numero de duplicados es minimo en comparación al volumen total de los datos entonces podemos eliminarlos sin que afecte nuestro análisis**

```
# Eliminamos los duplicados
logs.drop_duplicates(inplace=True)
logs.reset_index(inplace= True, drop = True)
```

Fecha

```
logs['full_date'] = pd.to_datetime(logs['full_date'], unit = 's')
logs.head(2)
```

	event_name	user_id	full_date	group
0	MainScreenAppear	4575588528974610257	2019-07-25 04:43:36	246
1	MainScreenAppear	7416695313311560658	2019-07-25 11:11:42	246

Agregamos columnas de fecha y hora separadas de la fecha completa

```
logs['fecha'] = logs['full_date'].dt.date
logs['hora'] = logs['full_date'].dt.time
display(logs.head())
logs.info()
```

	event_name	user_id	full_date	group	fecha	hora
0	MainScreenAppear	4575588528974610257	2019-07-25 04:43:36	246	2019-07-25	04:43:36
1	MainScreenAppear	7416695313311560658	2019-07-25 11:11:42	246	2019-07-25	11:11:42
2	PaymentScreenSuccessful	3518123091307005509	2019-07-25 11:28:47	248	2019-07-25	11:28:47
3	CartScreenAppear	3518123091307005509	2019-07-25 11:28:47	248	2019-07-25	11:28:47
4	PaymentScreenSuccessful	6217807653094995999	2019-07-25 11:48:42	248	2019-07-25	11:48:42

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243713 entries, 0 to 243712
Data columns (total 6 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   event_name  243713 non-null  category
 1   user_id     243713 non-null  int64
 2   full_date   243713 non-null  datetime64[ns]
 3   group       243713 non-null  category
 4   fecha       243713 non-null  object
 5   hora        243713 non-null  object
dtypes: category(2), datetime64[ns](1), int64(1), object(2)
memory usage: 7.9+ MB
```

```
logs['group'] = pd.to_numeric(logs['group']).astype(int)
```

Estudiar y comprobar los datos

```
# Cuántos eventos hay en los registros?
print(logs['event_name'].nunique())
print(logs['event_name'].unique())
```

```
5
['MainScreenAppear', 'PaymentScreenSuccessful', 'CartScreenAppear', 'OffersScreenAppe
ar', 'Tutorial']
Categories (5, object): ['CartScreenAppear', 'MainScreenAppear', 'OffersScreenAppea
r', 'PaymentScreenSuccessful', 'Tutorial']
```

```
# Cuántos usuarios hay en los registros?
logs['user_id'].nunique()
```

```
7551
```

```
# Cuál es el promedio de eventos por usuario?
logs.groupby('user_id')['event_name'].count().mean()
```

```
np.float64(32.27559263673685)
```

¿Qué periodo de tiempo cubren los datos? Encuentra la fecha máxima y mínima. Traza un histograma por fecha y hora. ¿Puedes tener seguridad de que tienes datos igualmente completos para todo el periodo? Los eventos más antiguos podrían terminar en los registros de algunos usuarios por razones técnicas y esto podría sesgar el panorama general. Encuentra el momento en el que los datos comienzan a estar completos e ignora la sección anterior. ¿Qué periodo representan realmente los datos?

```
# fecha máxima y mínima
print(logs['fecha'].min(), logs['fecha'].max())
```

```
2019-07-25 2019-08-07
```

Gráfico de lineas

```
# Agrupar los datos por fecha y contar la cantidad de usuarios
logs_agg = logs.groupby('fecha')['user_id'].count().reset_index()

# Crear el gráfico de línea con Plotly Express
fig = px.line(logs_agg, x='fecha', y='user_id', title="Número de usuarios por fecha")

# Mostrar el gráfico
fig.show()
```

Crear el Histograma

Encontrar el Punto donde los Datos se Estabilizan

```
# Crear histograma de eventos por fecha y hora
fig = px.histogram(logs, x="full_date", title="Distribución de eventos por fecha y ho

# Mostrar el gráfico
fig.show()
```

Conclusión:

- El 31 de julio hubo un leve aumento en los usuarios pero el dia que en realidad hubo un aumento considerable de usuarios fue el día 1 de agosto de 2019

```
logs_filtrados = logs[logs['fecha'] > pd.to_datetime('2019-07-31').date()]
```

Asegurar la fiabilidad de los datos

```
# Revisar si perdímos muchos datos despues de la limpieza inicial
logs_filtrados['user_id'].nunique() / logs['user_id'].nunique()
```

```
0.9977486425638988
```

```
# Tenemos datos suficientes en todos los grupos?
```

```
logs_filtrados.groupby('group')['user_id'].nunique().reset_index()
```

	group	user_id
0	246	2484
1	247	2513
2	248	2537

Confirmar con un Nuevo Histograma

Después de filtrar los datos, podemos volver a hacer el histograma para asegurarnos de que los eventos están distribuidos de manera más uniforme:

```
fig = px.histogram(logs_filtrados, x="full_date", title="Histograma después de filtra
fig.show()
```

Conclusión:

- La distribución del histograma muestra que los eventos registrados se vuelven consistentes a partir de una fecha específica, lo que indica que antes de ese punto, los datos podrían estar incompletos o sesgados. Después de la limpieza, el periodo analizado representa un conjunto de datos más fiable y homogéneo, permitiendo conclusiones más precisas sobre el comportamiento de los usuarios.

Estudiar el embudo de eventos

Qué eventos hay en los registros y su frecuencia de suceso - Ordenados por frecuencia.

```
logs_filtrados.groupby('event_name')['user_id'].count().sort_values(ascending = False
```

```
/var/folders/yd/jr1h18jn6px95w7s343fr20h0000gp/T/ipykernel_4585/389334401.py:1: Futur
eWarning:

The default of observed=False is deprecated and will be changed to True in a future v
ersion of pandas. Pass observed=False to retain current behavior or observed=True to
adopt the future default and silence this warning.
```

	event_name	user_id
0	MainScreenAppear	117328
1	OffersScreenAppear	46333
2	CartScreenAppear	42303
3	PaymentScreenSuccessful	33918
4	Tutorial	1005

Cantidad de usuarios únicos que llegaron a cada evento:

```
logs_filtrados.groupby('event_name')['user_id'].nunique().sort_values(ascending = Fals
```

```
/var/folders/yd/jr1h18jn6px95w7s343fr20h0000gp/T/ipykernel_4585/2643568699.py:1: Futu
reWarning:

The default of observed=False is deprecated and will be changed to True in a future v
ersion of pandas. Pass observed=False to retain current behavior or observed=True to
adopt the future default and silence this warning.
```

	event_name	user_id
0	MainScreenAppear	7419
1	OffersScreenAppear	4593
2	CartScreenAppear	3734
3	PaymentScreenSuccessful	3539
4	Tutorial	840

Encontrar la cantidad de usuarios que realizaron cada una de estas acciones

- Calcular la proporción de usuarios que realizaron la acción al menos una vez.

```
user_by_event = logs_filtrados.groupby('event_name')['user_id'].nunique().sort_values(
user_by_event['users_in_ps'] = user_by_event['user_id'].shift(1)
user_by_event['cr_ps'] = user_by_event['user_id'] / user_by_event['users_in_ps']
user_by_event['dropoff_rate'] = 1 - user_by_event['cr_ps']

user_by_event
```

```
/var/folders/yd/jr1h18jn6px95w7s343fr20h0000gp/T/ipykernel_4585/70769747.py:1: Future
Warning:

The default of observed=False is deprecated and will be changed to True in a future v
ersion of pandas. Pass observed=False to retain current behavior or observed=True to
adopt the future default and silence this warning.
```

	event_name	user_id	users_in_ps	cr_ps	dropoff_rate
0	MainScreenAppear	7419	NaN	NaN	NaN
1	OffersScreenAppear	4593	7419.0	0.619086	0.380914
2	CartScreenAppear	3734	4593.0	0.812976	0.187024
3	PaymentScreenSuccessful	3539	3734.0	0.947777	0.052223
4	Tutorial	840	3539.0	0.237355	0.762645

Embudo:

```
fig = go.Figure(go.Funnel(
    y = ['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear','PaymentScreenSu
    x = [7419, 4593, 3734, 3539 ],
    textinfo = "value+percent initial"
))

fig.show()
```

¿En qué etapa pierdes más usuarios?

- En el evento OffersScreenAppear es cuando se pierde significativamente un mayor porcentaje de usuarios.

¿Qué porcentaje de usuarios hace todo el viaje desde su primer evento hasta el pago?

- Un 48% de usuarios realizan el pago (etapa final) desde que el primer evento.

Estudiar los resultados del experimento - Test A/B

```
logs_filtrados.groupby('group')['user_id'].nunique()
```

```
group
246    2484
247    2513
248    2537
Name: user_id, dtype: int64
```

```
base_conversion = logs_filtrados[['user_id','group']].drop_duplicates()
base_conversion
```

	user_id	group
2826	3737462046622621720	246
2830	1433840883824088890	247
2831	4899590676214355127	247
2838	1182179323890311443	246
2842	4613461174774205834	248
...	...	...
242926	5811573131275421338	248
243129	5365227480683749189	248
243364	6660805781687343085	246
243407	7823752606740475984	246
243449	3454683894921357834	247

7534 rows × 2 columns

```
converted = pd.DataFrame(data = {
    'user_id': logs_filtrados[logs_filtrados['event_name'] == 'PaymentScreenSuccessfu
    'converted': 1
})
converted
```

	user_id	converted
0	4613461174774205834	1
1	2712290788139738557	1
2	6049698452889664846	1
3	5653442602434498252	1
4	6126676435667432321	1
...	...	...
3534	4369662623769092250	1
3535	4876403292056911122	1
3536	1309234519709630135	1
3537	7823752606740475984	1
3538	4164287718073415198	1

3539 rows × 2 columns

```
payment_conversion = base_conversion.merge(converted, on ='user_id', how = 'left')
payment_conversion['converted'] = payment_conversion['converted'].fillna(0)
```

```
payment_conversion.head()
```

	user_id	group	converted
0	3737462046622621720	246	1.0
1	1433840883824088890	247	0.0
2	4899590676214355127	247	1.0
3	1182179323890311443	246	0.0
4	4613461174774205834	248	1.0

Prueba de diferencia de proporciones entre los grupos de control (246 vs. 247)

Objetivo: Comprobar si la tasa de conversión entre los grupos 246 y 247 es estadísticamente diferente.

Prueba estadística recomendada: Prueba Z para proporciones

Hipótesis:

- H0: la tasa promedio de conversion entre los grupos 246 y 247 son iguales.
- HA: la tasa promedio de conversion entre los grupos son distintas:

Fijaremos nuestro nivel de significancia $\alpha = 0.1$

```
group_246 = payment_conversion[payment_conversion['group'] == 246]['converted']
group_247 = payment_conversion[payment_conversion['group'] == 247]['converted']
```

```
from statsmodels.stats.proportion import proportions_ztest
from scipy.stats import chisquare

# Filtrar solo grupos 246 y 247
payment_conversion = payment_conversion[payment_conversion['group'].isin([246, 247])]

# Extraer conversiones y tamaños de muestra
group_246 = payment_conversion[payment_conversion['group'] == 246]['converted']
group_247 = payment_conversion[payment_conversion['group'] == 247]['converted']

successes = [group_246.sum(), group_247.sum()]
trials = [len(group_246), len(group_247)]

# Verificar suposiciones
if min(successes) < 5 or min([t - s for s, t in zip(successes, trials)]) < 5:
    print("Algunas celdas tienen menos de 5 observaciones, los resultados pueden no se

# Prueba de diferencia de proporciones
_, pvalor = proportions_ztest(successes, trials, alternative='two-sided')

# Evaluación del p-valor
```

```
alpha = 0.1
if pvalor < alpha:
    print('Rechazamos H0')
else:
    print('No rechazamos H0')
```

```
No rechazamos H0
```

Conclusión:

- Dado que el resultado de la prueba fue "No rechazamos H_0", podemos concluir que no hay evidencia estadística suficiente para afirmar que las tasas de conversión entre los grupos 246 y 247 son significativamente diferentes.
- Esto indica que, en nuestro test A/A, los grupos se comportan de manera similar, lo cual es un buen indicador de que la asignación aleatoria y el proceso de medición funcionan correctamente.

Evaluación de la distribución equitativa de usuarios en los grupos A/A

Objetivo: Verificar que los grupos 246 y 247 están distribuidos de manera uniforme.

Prueba estadística recomendada: Prueba de Chi-Cuadrado

- H0: Los usuarios están distribuidos equitativamente entre los grupos 246 y 247.
- H1: Hay una diferencia en la distribución de los usuarios entre los grupos.

Si **p_value < 0.1**, los grupos no están equilibrados.

```
# Evaluación de la distribución equitativa de usuarios en los grupos A/A
observed = [logs_filtrados[logs_filtrados['group'] == 246]['user_id'].nunique(),
            logs_filtrados[logs_filtrados['group'] == 247]['user_id'].nunique()]
expected = [sum(observed)/2, sum(observed)/2]

_, pvalor = chisquare(observed, expected)
if pvalor < alpha:
    print('Rechazamos H0')
else:
    print('No rechazamos H0')
```

```
No rechazamos H0
```

Conclusión:

- Dado que el resultado de la prueba de Chi-Cuadrado fue "No rechazamos H_0", podemos concluir que no hay una diferencia estadísticamente significativa en la distribución de usuarios entre los grupos 246 y 247.
- Esto indica que los grupos fueron asignados de manera equitativa, lo que es clave en un test A/A para garantizar que cualquier diferencia observada en futuras pruebas no se deba a un sesgo en la asignación de usuarios.

Comparación de la proporción del evento más popular en los grupos de control

Objetivo: Determinar si la proporción de usuarios que realizaron el evento más popular difiere significativamente entre los grupos de control (246 y 247).

Prueba estadística recomendada: Prueba Z para proporciones

- H0: La proporción del evento más popular es la misma en ambos grupos.
- H1: La proporción del evento más popular es diferente entre los grupos.

Si **p_value < 0.1**, hay diferencia significativa en la frecuencia del evento.

```
# Comparación de la proporción del evento más popular en los grupos de control
event_popular = logs_filtrados['event_name'].value_counts().idxmax()
event_counts = logs_filtrados[logs_filtrados['event_name'] == event_popular].groupby(

successes = successes = [
    event_counts.loc[246] if 246 in event_counts.index else 0,
    event_counts.loc[247] if 247 in event_counts.index else 0
]

trials = [logs_filtrados[logs_filtrados['group'] == 246]['user_id'].nunique(),
          logs_filtrados[logs_filtrados['group'] == 247]['user_id'].nunique()]

_, pvalor = proportions_ztest(successes, trials)
if pvalor < alpha:
    print('Rechazamos H0')
else:
    print('No rechazamos H0')
```

```
No rechazamos H0
```

Automatización para todos los eventos:

```
def test_proportions_by_event(df, group1, group2):
    results = {}
    for event in df['event_name'].unique():
        event_counts = df[df['event_name'] == event].groupby('group')['user_id'].nunic

        if group1 in event_counts and group2 in event_counts:
            successes = [event_counts[group1], event_counts[group2]]
            trials = [df[df['group'] == group1]['user_id'].nunique(),
                      df[df['group'] == group2]['user_id'].nunique()]

            z_stat, p_value = proportions_ztest(successes, trials)
            results[event] = (z_stat, p_value)

    return results

event_results = test_proportions_by_event(logs_filtrados, 246, 247)
```

```
for event, (z, p) in event_results.items():
    print(f"Evento: {event} - Prueba Z: estadístico {z}, valor p {p}")
```

```
Evento: Tutorial - Prueba Z: estadístico -0.0781614596617372, valor p 0.9376996189257
114
Evento: MainScreenAppear - Prueba Z: estadístico 0.3093441851418436, valor p 0.757059
7232046099
Evento: OffersScreenAppear - Prueba Z: estadístico 1.154987713610383, valor p 0.24809
54578522181
Evento: CartScreenAppear - Prueba Z: estadístico 1.203368576486285, valor p 0.2288337
2237997213
Evento: PaymentScreenSuccessful - Prueba Z: estadístico 1.5779948491596598, valor p
0.11456679313141847
```

Conclusión:

- Dado que en todos los eventos el valor p fue mayor que 0.1, no rechazamos la hipótesis nulaLa distribución de los eventos es similar entre los grupos de control → No hay evidencia de que la asignación a los grupos haya generado diferencias en la interacción de los usuarios con los eventos.

¿Qué significa esto?

- La distribución de los eventos es similar entre los grupos de control, No hay evidencia de que la asignación a los grupos haya generado diferencias en la interacción de los usuarios con los eventos.
- Los valores p de todos los eventos están por encima de 0.1, lo que refuerza la idea de que los grupos fueron divididos correctamente y se comportan de manera similar.
- Esto valida que los grupos A/A están balanceados y que cualquier diferencia futura con el grupo experimental podría deberse al efecto del experimento y no a un sesgo en la asignación.

Comparación del grupo de fuentes alteradas (248) con los grupos de control

Objetivo: Evaluar si la proporción de conversión en el grupo 248 es diferente a la de los grupos de control.

Prueba estadística recomendada: Prueba Z para proporciones

- H0: La tasa de conversión en el grupo 248 es igual a la de los grupos de control combinados.
- H1: La tasa de conversión en el grupo 248 es diferente.

Si **p_value < 0.1**, hay diferencia en la conversión.

```
# Comparación del grupo de fuentes alteradas (248) con los grupos de control
group_248 = payment_conversion[payment_conversion['group'] == 248]['converted']
group_control = payment_conversion[payment_conversion['group'].isin([246, 247])]['conv

successes = [group_248.sum(), group_control.sum()]
trials = [len(group_248), len(group_control)]
```

```
_, pvalor = proportions_ztest(successes, trials)
if pvalor < alpha:
    print('Rechazamos H0')
else:
    print('No rechazamos H0')
```

```
No rechazamos H0
```

```
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages/stats
models/stats/proportion.py:1004: RuntimeWarning:

invalid value encountered in divide

/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages/stats
models/stats/proportion.py:1020: RuntimeWarning:

divide by zero encountered in divide
```

Conclusión:

- El resultado nos dice que no hay evidencia suficiente para afirmar que la tasa de conversión en el grupo 248 sea diferente de la de los grupos de control combinados.
- El cambio en las fuentes tipográficas (grupo 248) no afectó significativamente la conversión, Los usuarios interactuaron y convirtieron de manera similar a los grupos de control (246 y 247).
- El p-valor no es menor a 0.1, lo que indica que cualquier diferencia observada en la conversión probablemente se deba al azar y no a un efecto real del cambio en la fuente.
- Esto sugiere que la modificación en la tipografía no impactó en la experiencia del usuario de manera relevante en términos de conversión.

Corrección por múltiples pruebas (Bonferroni)

Como hemos realizado múltiples pruebas de hipótesis, aplicamos la corrección de Bonferroni para evitar falsos positivos.

Método: Ajustamos el nivel de significancia dividiéndolo entre el número total de pruebas.

```
# Verificar si hay eventos con valores cero antes de hacer la prueba
def test_proportions_with_correction(df, group1, group2, alpha):
    event_counts = df.groupby(['event_name', 'group'])['user_id'].nunique().unstack()

    num_tests = len(event_counts)  # Número total de pruebas realizadas
    alpha_adjusted = alpha / num_tests  # Prueba Bonferroni
    print(f"Nuevo nivel de significancia ajustado: {alpha_adjusted}")

    for event in event_counts.index:
        if group1 not in event_counts.columns or group2 not in event_counts.columns:
            print(f"Evento: {event} - No se puede calcular p-valor debido a valores fa
            continue
```

```
        successes = [
            event_counts.loc[event, group1] if not pd.isna(event_counts.loc[event, gr
            event_counts.loc[event, group2] if not pd.isna(event_counts.loc[event, gr
        ]

        trials = [
            df[df['group'] == group1]['user_id'].nunique(),
            df[df['group'] == group2]['user_id'].nunique()
        ]

        if 0 in trials or 0 in successes:
            print(f"Evento: {event} - No se puede calcular p-valor debido a valores ce
            continue

        _, pvalor = proportions_ztest(successes, trials)
        if pvalor < alpha_adjusted:
            print(f"Evento: {event} - Diferencia significativa con p={pvalor:.5f} (aju
        else:
            print(f"Evento: {event} - No significativo con p={pvalor:.5f} (ajustado)")

# Ejecutamos la función con los grupos 246 y 247 y alfa ajustado
test_proportions_with_correction(logs_filtrados, 246, 247, 0.1)
```

```
Nuevo nivel de significancia ajustado: 0.02
Evento: CartScreenAppear - No significativo con p=0.22883 (ajustado)
Evento: MainScreenAppear - No significativo con p=0.75706 (ajustado)
Evento: OffersScreenAppear - No significativo con p=0.24810 (ajustado)
/var/folders/yd/jr1h18jn6px95w7s343fr20h0000gp/T/ipykernel_4585/538203962.py:3: Futur
eWarning:

The default of observed=False is deprecated and will be changed to True in a future v
ersion of pandas. Pass observed=False to retain current behavior or observed=True to
adopt the future default and silence this warning.

Evento: PaymentScreenSuccessful - No significativo con p=0.11457 (ajustado)
Evento: Tutorial - No significativo con p=0.93770 (ajustado)
```

Conclusión:

No hay diferencias significativas en ninguno de los eventos

- Todos los eventos tienen un p-valor mayor que el nivel de significancia ajustado (0.02).
- Esto significa que no encontramos evidencia suficiente para rechazar la hipótesis nula.
- En términos prácticos, la tasa de conversión en estos eventos es estadísticamente similar entre los grupos comparados.

El ajuste de Bonferroni hizo que la prueba fuera más estricta

- Antes, el nivel de significancia era 0.1, lo que permitía detectar más diferencias.
- Ahora, con 0.02, se reduce la probabilidad de falsos positivos, pero también se hace más difícil detectar efectos reales.

No hay evidencia de que los grupos tengan diferencias significativas en la proporción de eventos analizados.

Conclusión Final del Proyecto

Este análisis A/B tuvo como objetivo evaluar si la proporción de usuarios que realizaron ciertos eventos difería significativamente entre los grupos de control (246 y 247) y el grupo con fuentes alteradas (248). Para ello, utilizamos pruebas Z para proporciones y aplicamos la corrección de Bonferroni para ajustar el nivel de significancia debido a la realización de múltiples pruebas.

Principales Hallazgos:

Los grupos de control (246 y 247) no mostraron diferencias significativas entre sí.

- No encontramos evidencia estadística de que la proporción de usuarios que realizaron el evento más popular difiera entre ambos grupos.

El grupo con fuentes alteradas (248) no mostró diferencias significativas respecto a los grupos de control combinados.

- La tasa de conversión en el grupo 248 fue similar a la de los controles, lo que sugiere que la alteración en las fuentes no tuvo un impacto relevante en los eventos analizados.

Conclusión Final:

- No se encontraron diferencias significativas en la proporción de usuarios que realizaron los eventos analizados entre los grupos comparados.
- El grupo con fuentes alteradas no mostró un impacto relevante en la conversión respecto a los grupos de control.
- Las fuentes de tráfico utilizadas en el experimento no parecen influir en la interacción de los usuarios con la plataforma.