e3b255f4-1c60-4029-81c8-99c9f6f53d9d-Copy1

November 24, 2024

1 Proyecto Final CallMeMaybe

1. Introduccion
2. Descomposicion
3. Analisis exploratorio de datos

- 3.1 Operadores con un alto porcentaje de llamadas perdidas (tanto internas como externas)
- 3.1.1 Hipótesis 1: Los operadores con una alta tasa de llamadas entrantes perdidas tienen un menor desempeño
- 3.2 Tiempo de espera promedio
- 3.2.1 Hipotesis 2: Existe una diferencia significativa en el tiempo de espera promedio entre los operadores "lentos" y los "rápidos"
- 3.3 Número de llamadas salientes por operador

4. Conclusion y recomendaciones
5. Link para presentacion y dashboard

1.1 Introducción

En un entorno altamente competitivo como el de las telecomunicaciones, la calidad de la atención al cliente es un factor determinante para el éxito de una empresa. CallMeMaybe, consciente de esta realidad, busca constantemente mejorar la eficiencia y la calidad de su servicio. Este estudio tiene como objetivo analizar el desempeño de los operadores de la compañía durante el período comprendido entre 01/08/2019 y 27/11/2019, con el fin de identificar aquellos que no cumplen con los estándares establecidos en términos de tiempo de respuesta, resolución de incidencias y satisfacción del cliente.

A través de un análisis exhaustivo de los datos históricos de llamadas, se calcularán métricas clave como el porcentaje de llamadas perdidas, el tiempo de espera promedio y el número de llamadas resueltas por operador. Además, se realizarán pruebas estadísticas para evaluar si existen diferencias significativas en el desempeño entre los diferentes grupos de operadores. Los resultados de este estudio permitirán identificar las áreas de mejora, diseñar programas de capacitación más efectivos y tomar decisiones estratégicas para optimizar la asignación de recursos y mejorar la experiencia del cliente.

1.2 Descomposicion

1.Objetivo ¿Qué queremos conseguir y por qué? Identificar a los operadores que no están cumpliendo con los estándares de servicio de CallMeMaybe, medidos por métricas como el número de llamadas perdidas y el tiempo de espera. Esto permitirá tomar acciones correctivas, mejorar la

calidad del servicio y aumentar la satisfacción del cliente. ¿A quién le interesa lo que produces? Los supervisores de CallMeMaybe, los clientes y el equipo de gestión de la compañía. ¿Qué decisiones se tomarán de acuerdo al análisis? Se podrán tomar decisiones sobre capacitación adicional para los operadores, reasignación de llamadas, ajustes en los planes tarifarios o incluso la eliminación de operadores que de manera consistente no cumplan con los estándares.

Especificar los detalles Operador ineficaz: Un operador será considerado ineficaz si:

Tiene un alto porcentaje de llamadas perdidas (tanto internas como externas) por encima del 30%.

Tiene un tiempo de espera promedio alto para las llamadas entrantes (por encima de 235 segundos).

Tiene un bajo número de llamadas salientes en comparación con otros operadores con perfiles similares (menos de 12 llamadas).

Proponer hipótesis Hipótesis 1: Los operadores con una alta tasa de llamadas entrantes perdidas tienen un menor desempeño.

Hipótesis 2: Existe una diferencia significativa en el tiempo de espera promedio entre los operadores "lentos" y los "rápidos"

Convertir las hipótesis en un plan de acción claro *Análisis Exploratorio de Datos Limpiar los datos:Verificar la consistencia de los datos (tipos de datos, valores faltantes, outliers). Unificar formatos de fechas y horas.

*Explorar los datos: Calcular estadísticas descriptivas para cada variable (media, mediana, desviación estándar). Identificar correlaciones entre variables (matriz de correlación). Identificar operadores ineficaces

*Calcular métricas clave: Calcular la tasa de llamadas perdidas, tiempo de espera promedio y número de llamadas salientes por operador. Crear una clasificación de los operadores en función de estas métricas.

*Visualizar resultados: Crear gráficos de barras para comparar el desempeño de los operadores.

Prueba de hipótesis estadísticas

*Seleccionar pruebas estadísticas: Utilizar correlación de Pearson para medir la relación entre variables numéricas. Utilizar Mann-Whitney U cuando la distribucion no sea normal.

*Interpretar resultados: Evaluar si los resultados obtenidos respaldan o rechazan las hipótesis planteadas.

Plan de acción detallado Importar los datos.

Limpiar los datos: Convertir columnas de fecha a formato datetime. Imputar o reemplazar valores faltantes y atipicos.

Calcular métricas: Agrupar los datos por operador y calcular la tasa de llamadas perdidas, tiempo de espera promedio y número de llamadas salientes.

Visualizar los resultados: Crear un gráfico de barras para comparar la tasa de llamadas perdidas de cada operador. Crear un boxplot para visualizar la distribución del tiempo de espera. Crear un scatter plot para explorar la relación entre el número de llamadas salientes y la tasa de llamadas perdidas.

Identificar operadores ineficaces: Establecer umbrales para cada métrica. Identificar los operadores que superan estos umbrales.

Prueba de hipótesis: Verificar la normalidad de los datos mediante prueba Shapiro-Wilk

Mediante prueba no parametrica (Mann-Whitney U)verificar que los operadores con una alta tasa de llamadas entrantes perdidas tienen un menor desempeño. Y si existe una diferencia significativa en el tiempo de espera promedio entre los operadores "lentos" y los "rápidos". Observar que cantidad de operadores tienen pocas llamadas salientes

1.3 Analisis exploratorio de datos

[1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pytz
from scipy import stats
from scipy.stats import shapiro, mannwhitneyu
```

[2]:
```
df_llamadas = pd.read_csv ("/datasets/telecom_dataset_us.csv")
df_clientes = pd.read_csv("/datasets/telecom_clients_us.csv")
```

[3]:
```
df_llamadas['date'] = pd.to_datetime(df_llamadas['date'], format='%Y-%m-%d')
df_llamadas['date'] = pd.to_datetime(df_llamadas['date'], format='%Y-%m-%d')

# Convertir a la zona horaria UTC
df_llamadas['date'] = df_llamadas['date'].dt.tz_convert('UTC')

df_llamadas['fecha'] = df_llamadas['date'].dt.date
df_llamadas['fecha'] = pd.to_datetime(df_llamadas['fecha'], format='%Y-%m-%d')
df_llamadas.drop('date', axis=1, inplace=True)
df_llamadas.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53902 entries, 0 to 53901
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   user_id              53902 non-null  int64
 1   direction            53902 non-null  object
 2   internal             53785 non-null  object
 3   operator_id          45730 non-null  float64
 4   is_missed_call       53902 non-null  bool
 5   calls_count          53902 non-null  int64
 6   call_duration        53902 non-null  int64
 7   total_call_duration  53902 non-null  int64
 8   fecha                53902 non-null  datetime64[ns]
```

```
dtypes: bool(1), datetime64[ns](1), float64(1), int64(4), object(2)
memory usage: 3.3+ MB
```

[4]:
```
df_llamadas['operator_id'] = df_llamadas['operator_id'].replace(' ', np.nan)
df_llamadas.isnull().sum()
```

[4]:
```
user_id                 0
direction               0
internal              117
operator_id          8172
is_missed_call          0
calls_count             0
call_duration           0
total_call_duration     0
fecha                   0
dtype: int64
```

[5]:
```
df_llamadas['internal'] = df_llamadas['internal'].fillna(False)

df_llamadas['operator_id'] = df_llamadas['operator_id'].fillna('desconocido')
```

[6]:
```
df_llamadas.head()
```

[6]:

	user_id	direction	internal	operator_id	is_missed_call	calls_count	call_duration	total_call_duration	fecha
0	166377	in	False	desconocido	True	2	0	4	2019-08-03
1	166377	out	True	880022.0	True	3	0	5	2019-08-04
2	166377	out	True	880020.0	True	1	0	1	2019-08-04
3	166377	out	True	880020.0	False	1	10	18	2019-08-04
4	166377	out	False	880022.0	True	3	0	25	2019-08-04

[7]:
```
# Calculando estadísticas descriptivas
df_llamadas.describe()
```

[7]:

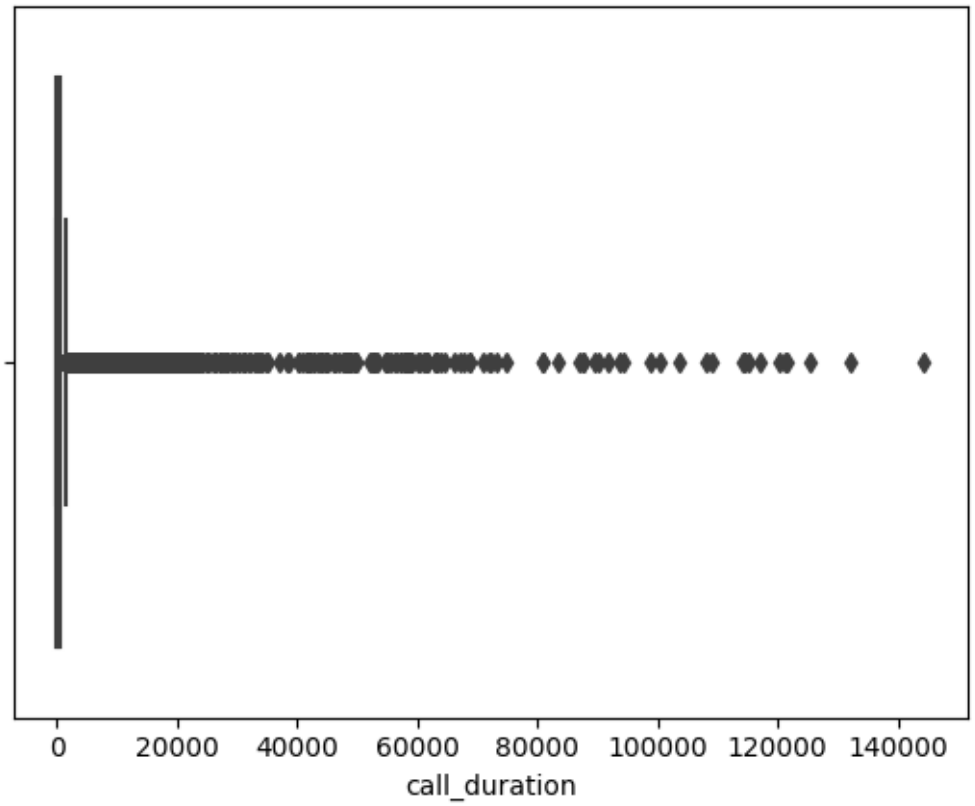
	user_id	calls_count	call_duration	total_call_duration
count	53902.000000	53902.000000	53902.000000	53902.000000
mean	167295.344477	16.451245	866.684427	1157.133297
std	598.883775	62.917170	3731.791202	4403.468763
min	166377.000000	1.000000	0.000000	0.000000
25%	166782.000000	1.000000	0.000000	47.000000
50%	167162.000000	4.000000	38.000000	210.000000

```
75%     167819.000000     12.000000     572.000000           902.000000
max     168606.000000   4817.000000  144395.000000        166155.000000
```

Existe una gran variabilidad en el número de llamadas, la duración de las llamadas y la duración total de las llamadas por usuario. Esto sugiere que tenemos una población de usuarios bastante heterogénea en términos de su comportamiento de llamadas.

Posibles outliers, los valores mínimos y máximos de call_duration y total_call_duration sugieren la presencia de outliers (valores atípicos).

[8]:
```
# Crear un boxplot para visualizar outliers en la columna 'call_duration'
sns.boxplot(x='call_duration', data=df_llamadas)
plt.show()

# Calcular el IQR y los límites para identificar outliers
Q1 = df_llamadas['call_duration'].quantile(0.25)
Q3 = df_llamadas['call_duration'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Identificar los outliers
outliers = df_llamadas[(df_llamadas['call_duration'] < lower_bound) |␣
 ↪(df_llamadas['call_duration'] > upper_bound)]
outliers
```

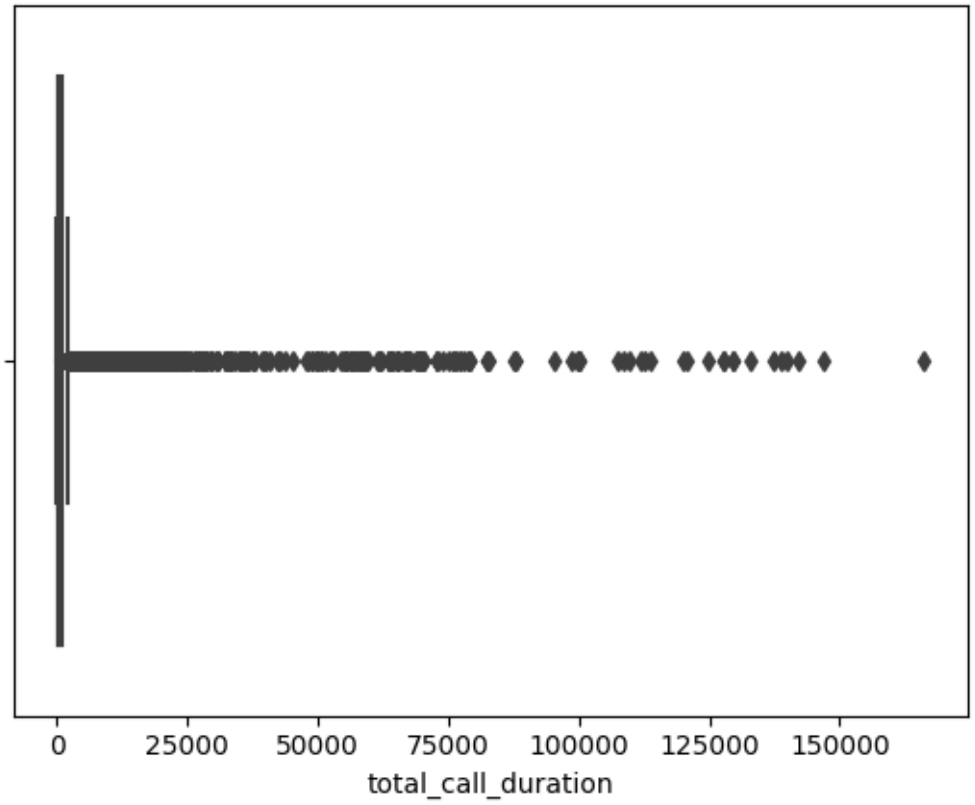


[8]:

	user_id	direction	internal	operator_id	is_missed_call	calls_count	call_duration	total_call_duration	fecha
21	166377	out	False	880028.0	False	17	1603	1725	2019-08-08
24	166377	out	False	880028.0	False	20	2074	2191	2019-08-11
41	166377	out	False	880028.0	False	18	2686	2782	2019-08-13
43	166377	out	False	880026.0	False	10	1567	1654	2019-08-13
44	166377	out	False	880026.0	False	10	1567	1654	2019-08-13
…	…	…	…	…	…	…	…	…	…
53848	168601	out	False	952914.0	False	46	2614	3221	2019-11-13
53850	168601	out	False	952914.0	False	17			
53865	168601	out	False	952914.0	False	30			
53899	168606	out	True	957922.0	False	4			
53900	168606	out	True	957922.0	False	4			

```
53850           1472                1705 2019-11-14
53865           1650                2119 2019-11-24
53899           3130                3190 2019-11-14
53900           3130                3190 2019-11-14

[7571 rows x 9 columns]
```

[9]:
```
# Crear un boxplot para visualizar outliers en la columna 'total_call_duration'
sns.boxplot(x='total_call_duration', data=df_llamadas)
plt.show()

# Calcular el IQR y los límites para identificar outliers
Q1 = df_llamadas['total_call_duration'].quantile(0.25)
Q3 = df_llamadas['total_call_duration'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Identificar los outliers
outliers = df_llamadas[(df_llamadas['total_call_duration'] < lower_bound) |␣
 ↪(df_llamadas['total_call_duration'] > upper_bound)]
```



[10]:
```
# Calcular los percentiles 95 para ambas columnas
percentil_95_call_duration = np.percentile(df_llamadas['call_duration'], 95)
percentil_95_total_call_duration = np.
 ↪percentile(df_llamadas['total_call_duration'], 95)

print(percentil_95_call_duration)
print(percentil_95_total_call_duration)
```

```
3739.949999999997
4540.0
```

[11]:
```
# Calcular el número de llamadas que superan el percentil 95 para cada columna
num_llamadas_sobre_percentil_95_call_duration =␣
 ↪df_llamadas[df_llamadas['call_duration'] > percentil_95_call_duration].
 ↪shape[0]
num_llamadas_sobre_percentil_95_total_call_duration =␣
 ↪df_llamadas[df_llamadas['total_call_duration'] >␣
 ↪percentil_95_total_call_duration].shape[0]

print("Número de llamadas con duración superior al percentil 95:",␣
 ↪num_llamadas_sobre_percentil_95_call_duration)
print("Número de llamadas con duración total superior al percentil 95:",␣
 ↪num_llamadas_sobre_percentil_95_total_call_duration)
```

```
Número de llamadas con duración superior al percentil 95: 2696
Número de llamadas con duración total superior al percentil 95: 2695
```

[12]:
```
# Calcular el número de llamadas que superan el percentil 95 para cada columna
num_llamadas_sobre_percentil_95_call_duration =␣
 ↪df_llamadas[df_llamadas['call_duration'] > percentil_95_call_duration].
 ↪shape[0]
print("Número de llamadas con duración superior al percentil 95:",␣
 ↪num_llamadas_sobre_percentil_95_call_duration)

# Crear una nueva columna para categorizar las llamadas
df_llamadas['tipo_llamada'] = np.where(df_llamadas['call_duration'] >␣
 ↪percentil_95_call_duration, 'Llamada larga', 'Llamada estándar')

# Visualizar las primeras filas para verificar
print(df_llamadas.head())
```

```
Número de llamadas con duración superior al percentil 95: 2696
   user_id direction  internal  operator_id  is_missed_call  calls_count  \
0   166377        in     False  desconocido            True            2
1   166377       out      True     880022.0            True            3
2   166377       out      True     880020.0            True            1
3   166377       out      True     880020.0           False            1
```

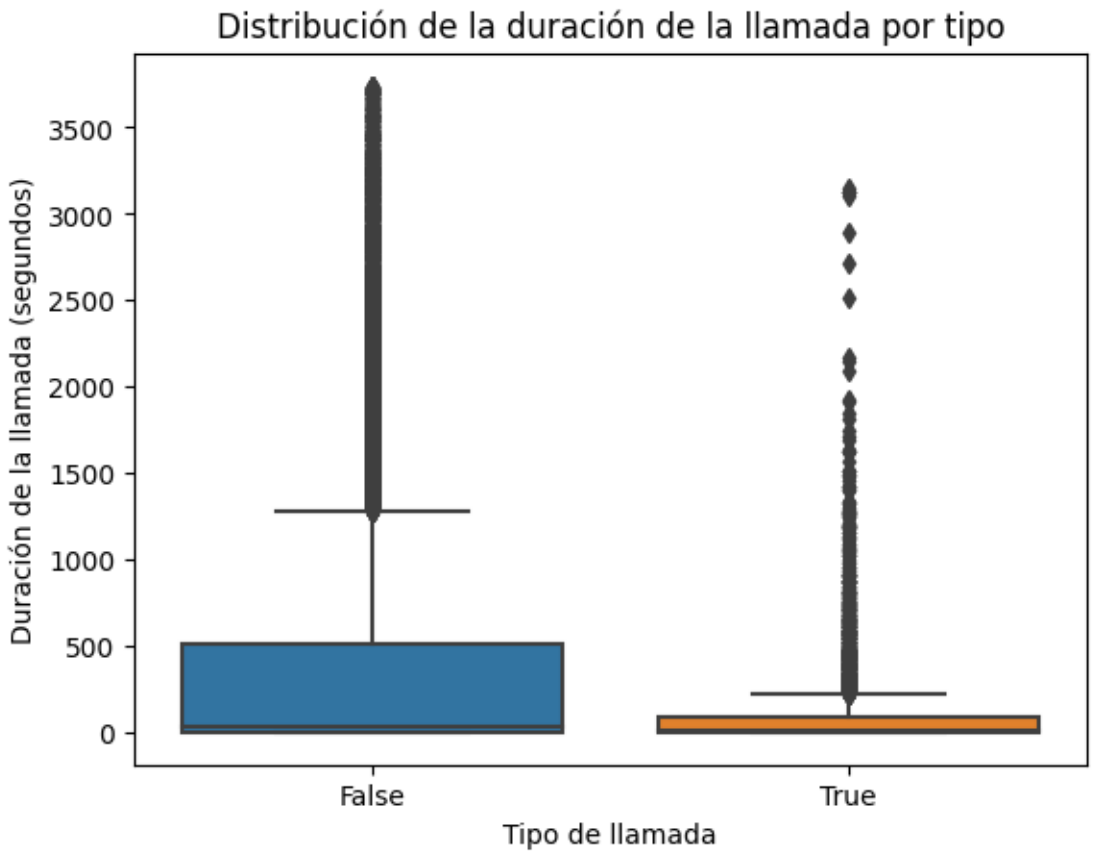
```
4   166377       out      False     880022.0            True            3
```

```
   call_duration  total_call_duration      fecha      tipo_llamada
0              0                    4 2019-08-03  Llamada estándar
1              0                    5 2019-08-04  Llamada estándar
2              0                    1 2019-08-04  Llamada estándar
3             10                   18 2019-08-04  Llamada estándar
4              0                   25 2019-08-04  Llamada estándar
```

Como la cantidad de llamadas por encima del percentil 95, es un numero considerable. Se creo una columna con la categoria de llamadas largas y asi usar estos datos en futuros analisis.

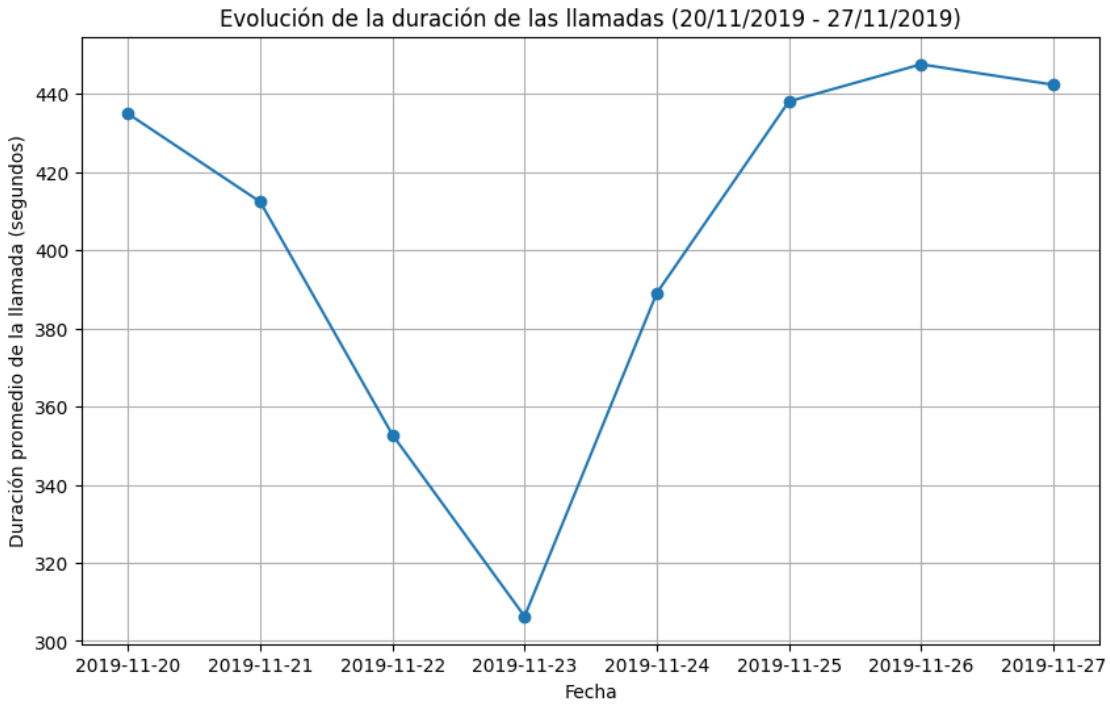
[13]:
```
# Crear un df sin los outliers
df_llamadas_2= df_llamadas[(df_llamadas['call_duration'] <=␣
 ↪percentil_95_call_duration) &
                                    (df_llamadas['total_call_duration'] <=␣
 ↪percentil_95_total_call_duration)]
```

[14]:
```
sns.boxplot(x='internal', y='call_duration', data=df_llamadas_2)
plt.xlabel('Tipo de llamada')
plt.ylabel('Duración de la llamada (segundos)')
plt.title('Distribución de la duración de la llamada por tipo')
plt.show()
```



Las llamadas internas presentan una duración significativamente mayor que las externas. Esto sugiere que las interacciones entre operadores y clientes suelen requerir más tiempo para resolverse.

[25]:
```
# Filtramos las llamadas de la semana del 20 al 27 de noviembre de 2019
df_semana = df_llamadas_2[(df_llamadas_2['fecha'] >= '2019-11-20') &␣
 ↪(df_llamadas_2['fecha'] <= '2019-11-27')]

# Agrupamos los datos por día y calculamos la duración promedio de las llamadas
df_agrupado = df_semana.groupby('fecha')['call_duration'].mean().reset_index()

# Creamos el gráfico de línea
plt.figure(figsize=(10, 6))
plt.plot(df_agrupado['fecha'], df_agrupado['call_duration'], marker='o')
plt.xlabel('Fecha')
plt.ylabel('Duración promedio de la llamada (segundos)')
plt.title('Evolución de la duración de las llamadas (20/11/2019 - 27/11/2019)')
plt.grid(True)
plt.show()
```

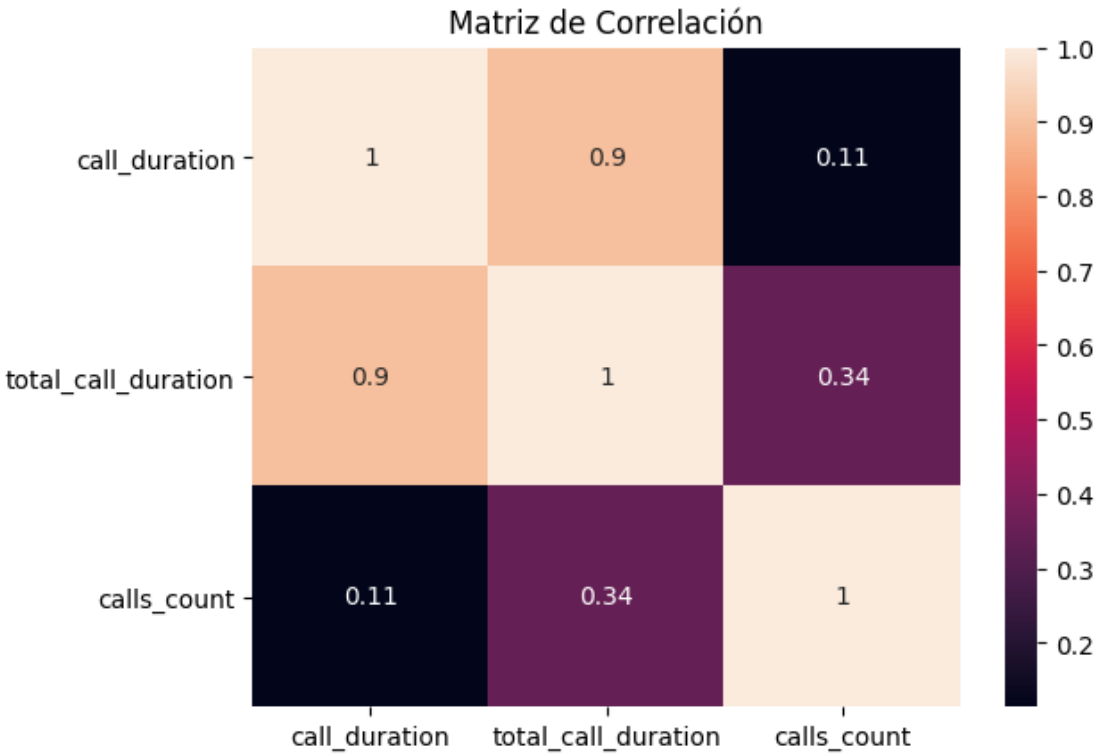


La duración promedio de las llamadas presentó un patrón interesante durante la semana del 20 al 27 de noviembre. Se observa una disminución inicial seguida de un incremento sostenido. Este comportamiento sugiere la posible influencia de factores externos, como promociones, campañas

o cambios en los procesos internos, que podrían haber afectado la complejidad y duración de las interacciones con los clientes.

La eliminación de los outliers en las columnas 'call_duration' y 'total_call_duration' ha resultado en una reducción significativa de la desviación estándar, lo que indica una menor dispersión de los datos. Al establecer un límite en el percentil 95 y eliminar los valores superiores, se ha logrado obtener una distribución de datos más homogénea, facilitando así el análisis y la interpretación de los resultados.

[20]:
```
# Seleccionar solo las columnas numéricas continuas relevantes
columnas_numericas = ['call_duration', 'total_call_duration', 'calls_count']
df_numericas_continuas = df_llamadas_2[columnas_numericas]

# Calcular la matriz de correlación
cm = df_numericas_continuas.corr()

# Visualizar la matriz de correlación
sns.heatmap(cm, annot=True)
plt.title("Matriz de Correlación")
plt.show()
```



La matriz de correlación muestra una fuerte relación positiva entre la duración de las llamadas individuales (call_duration) y la duración total de las llamadas (total_call_duration), lo que es esperable. Sin embargo, la relación entre el número total de llamadas (calls_count) y las otras dos

variables es más débil, sugiriendo que la eficiencia de los operadores podría influir en la duración total de las llamadas, independientemente del número de ellas. También podría sugerir que los usuarios están teniendo dificultades para resolver sus problemas en una sola llamada.

1.3.1 Operadores con un alto porcentaje de llamadas perdidas (tanto internas como externas)

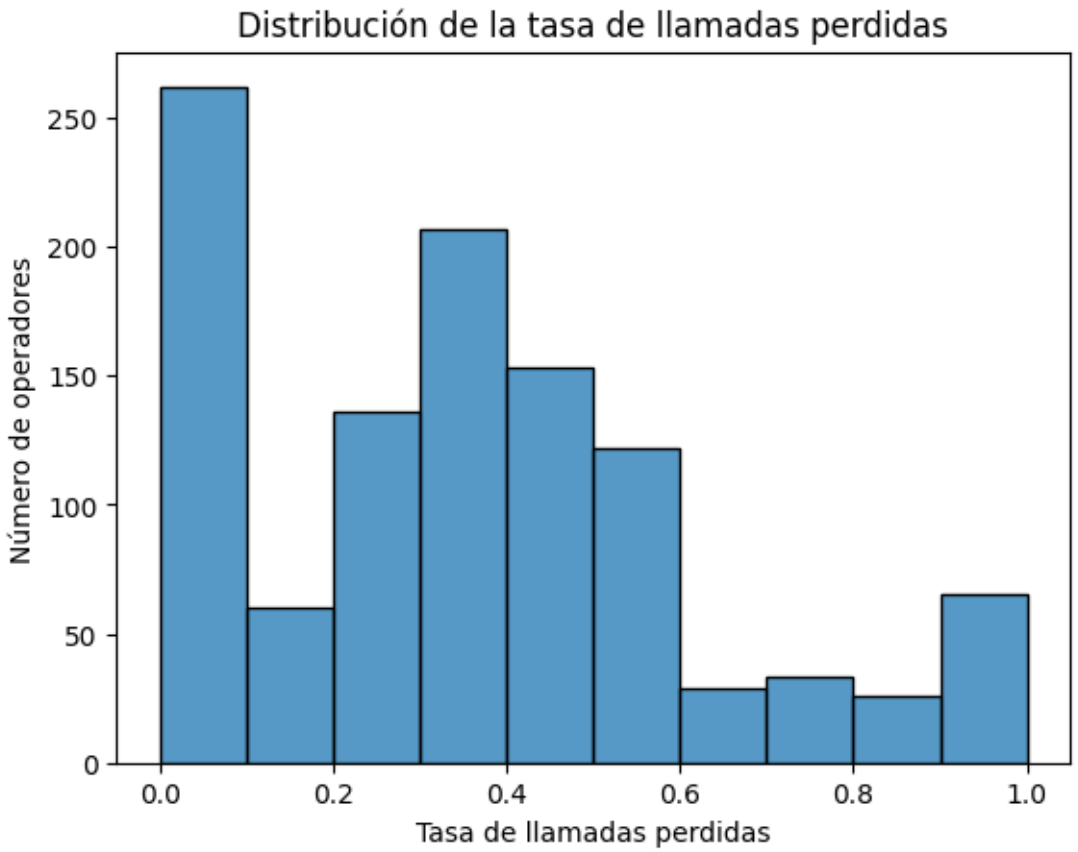
[22]:
```
# Agrupar por operador y calcular la tasa de llamadas perdidas
grouped_data = df_llamadas_2.groupby('operator_id').agg(␣
 ↪missed_call_rate=('is_missed_call', 'mean'))

# Ordenar por tasa de llamadas perdidas de mayor a menor
sorted_data = grouped_data.sort_values(by='missed_call_rate', ascending=False)

# Establecer umbral
llamadas_perdidas_limite= 0.30

# Identificar operadores ineficientes
inefficient_operators = sorted_data[sorted_data['missed_call_rate'] >␣
 ↪llamadas_perdidas_limite]

# Contar los operadores ineficientes
num_inefficient_operators = len(inefficient_operators)
print("Número de operadores con alto porcentaje de llamadas perdidas:",␣
 ↪num_inefficient_operators)

# Mostrar los operadores ineficientes
print(inefficient_operators)
```

```
Número de operadores con alto porcentaje de llamadas perdidas: 635
             missed_call_rate
operator_id
936296.0             1.000000
905480.0             1.000000
937772.0             1.000000
885682.0             1.000000
937770.0             1.000000
…                         …
962268.0             0.303030
925826.0             0.302521
937860.0             0.302326
900826.0             0.301676
906400.0             0.301205

[635 rows x 1 columns]
```

[23]:
```
df_llamadas_2['operator_id'].nunique()
```

[23]: 1093

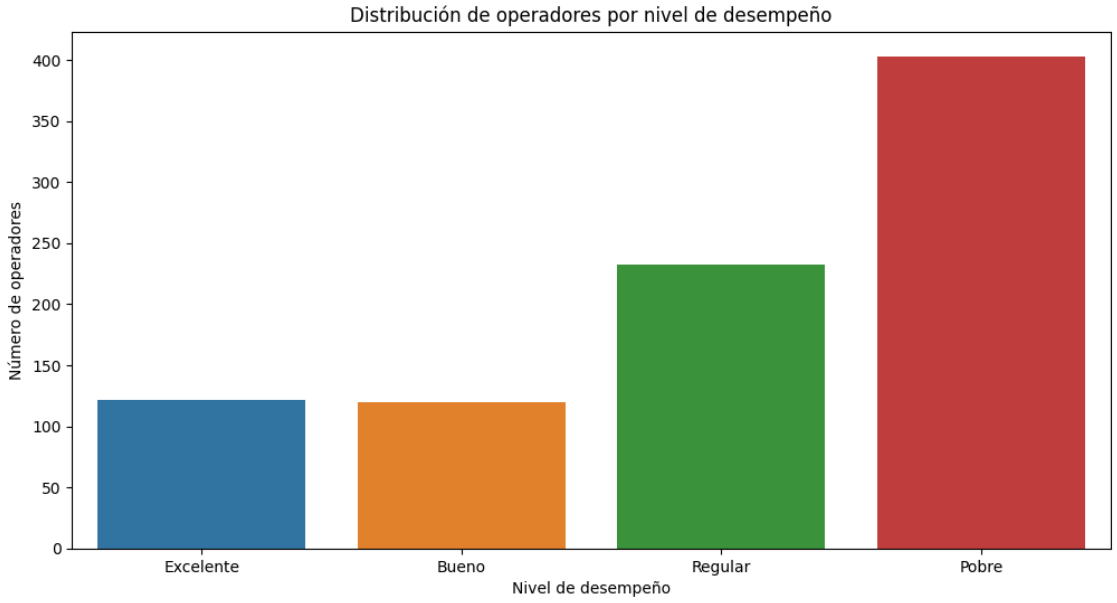
[24]:
```
# Histograma
sns.histplot(data=sorted_data, x='missed_call_rate', bins=10)
plt.title('Distribución de la tasa de llamadas perdidas')
plt.xlabel('Tasa de llamadas perdidas')
plt.ylabel('Número de operadores')
plt.show()
```



La mayoría de los operadores presentan una tasa de llamadas perdidas baja, concentrándose en el rango entre 0 y 0.4. Sin embargo, existe un grupo considerable de operadores con tasas más elevadas, lo que sugiere la necesidad de implementar acciones para mejorar la eficiencia en la atención al cliente.

[19]:
```
# Definir los rangos de desempeño para mejor visibilidad de patrones
bins = [0, 0.2, 0.3, 0.4, 1]
labels = ['Excelente', 'Bueno', 'Regular', 'Pobre']

# Crear una nueva columna con los rangos de desempeño
```

```
sorted_data['desempeño'] = pd.cut(sorted_data['missed_call_rate'], bins=bins,␣
 ↪labels=labels)

# Gráfico de barras
plt.figure(figsize=(12, 6))
sns.countplot(x='desempeño', data=sorted_data, order=labels)
plt.title('Distribución de operadores por nivel de desempeño')
plt.xlabel('Nivel de desempeño')
plt.ylabel('Número de operadores')
plt.show()
```



Si tenemos 1093 operadores en total y 635 tienen una tasa de llamadas perdidas superior al 30%, estamos hablando de aproximadamente el 58% del equipo. Esto es un porcentaje considerable y sugiere un problema sistémico más allá de casos individuales.

Impacto que representa para el negocio una tasa de llamadas perdidas tan alta puede tener un choque negativo significativo en la reputación de la empresa, la satisfacción del cliente y, en última instancia, en los ingresos. Los clientes que no pueden comunicarse con un agente pueden optar por buscar los servicios de la competencia.

El análisis de la tasa de llamadas perdidas por operador revela una amplia gama de desempeños. Si bien la mayoría de los operadores mantienen tasas bajas, un grupo significativo presenta tasas significativamente más altas. Es necesario investigar las causas de estas diferencias, como la experiencia del operador, la complejidad de las llamadas o problemas técnicos, para implementar acciones correctivas y garantizar un servicio de calidad

El gráfico de barras evidencia una clara necesidad de implementar programas de capacitación y desarrollo para los operadores que se encuentran en las categorías “Regular” y “Pobre”. Además, sería conveniente realizar un análisis más detallado de los factores que influyen en el desempeño de

cada operador, como la experiencia, la carga de trabajo o los recursos disponibles.

Hipótesis 1: Los operadores con una alta tasa de llamadas entrantes perdidas tienen un menor desempeño

[20]:
```
# Calcular métricas adicionales por operador
grouped_data = df_llamadas_2.groupby('operator_id').agg(
    missed_call_rate=('is_missed_call', 'mean'),
    total_calls=('calls_count', 'sum'),
    avg_call_duration=('call_duration', 'mean'))

# Dividir los datos en dos grupos
high_missed_calls = grouped_data[grouped_data['missed_call_rate'] > 0.3]
low_missed_calls = grouped_data[grouped_data['missed_call_rate'] <= 0.3]

# Verificar normalidad de la duración promedio de las llamadas en ambos grupos
# Grupo con alta tasa de llamadas perdidas
stat, p = stats.shapiro(high_missed_calls['avg_call_duration'])
print('Grupo con alta tasa de llamadas perdidas:')
print('Estadístico de Shapiro-Wilk:', stat)
print('p-valor:', p)
if p > 0.05:
    print('Los datos se distribuyen aproximadamente de forma normal')
else:
    print('Los datos no se distribuyen de forma normal')

print('------------------------------------------------------------')
# Grupo con baja tasa de llamadas perdidas
stat, p = stats.shapiro(low_missed_calls['avg_call_duration'])
print('Grupo con baja tasa de llamadas perdidas:')
print('Estadístico de Shapiro-Wilk:', stat)
print('p-valor:', p)
if p > 0.05:
    print('Los datos se distribuyen aproximadamente de forma normal')
else:
    print('Los datos no se distribuyen de forma normal')
```

```
Grupo con alta tasa de llamadas perdidas:
Estadístico de Shapiro-Wilk: 0.928991436958313
p-valor: 9.061032382856921e-17
Los datos no se distribuyen de forma normal
------------------------------------------------------------
Grupo con baja tasa de llamadas perdidas:
Estadístico de Shapiro-Wilk: 0.5778684616088867
p-valor: 8.375935626021475e-32
Los datos no se distribuyen de forma normal
```

[21]:
```
# Comparar el número total de llamadas atendidas (prueba de Mann-Whitney U)
U_stat, p_value = stats.mannwhitneyu(high_missed_calls['total_calls'],
                                     low_missed_calls['total_calls'])
print('Comparación del número total de llamadas atendidas:')
print('Estadístico U:', U_stat)
print('p-valor:', p_value)
print('-----------------------------------------------------------------')
# Comparar la duración promedio de las llamadas (prueba de Mann-Whitney U)
U_stat, p_value = stats.mannwhitneyu(high_missed_calls['avg_call_duration'],
                                     low_missed_calls['avg_call_duration'])
print('Comparación de la duración promedio de las llamadas:')
print('Estadístico U:', U_stat)
print('p-valor:', p_value)
```

```
Comparación del número total de llamadas atendidas:
Estadístico U: 220403.0
p-valor: 4.6814544238210916e-48
-----------------------------------------------------------------
Comparación de la duración promedio de las llamadas:
Estadístico U: 183130.5
p-valor: 2.393128334781647e-13
```

Los resultados obtenidos indican una relación clara y significativa entre la tasa de llamadas perdidas y el desempeño de los operadores. Los operadores con una alta tasa de llamadas perdidas tienen un menor volumen de trabajo y una menor duración promedio de las llamadas, lo que sugiere una necesidad de mejorar sus prácticas y procesos.

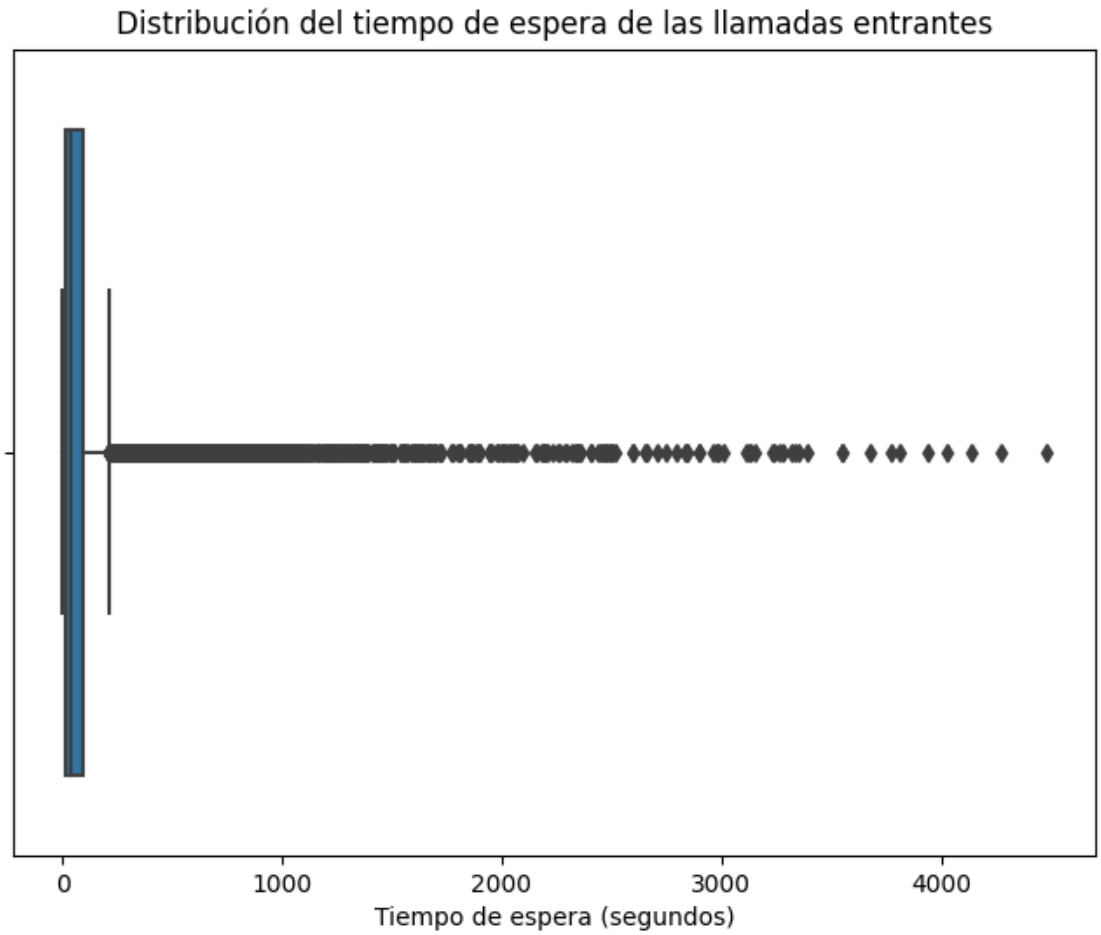
1.3.2 Tiempo de espera promedio

[22]:
```
# Filtrar las llamadas entrantes
llamadas_entrantes = df_llamadas_2[df_llamadas_2['direction'] == 'in']

# Calcular el tiempo total de espera
tiempo_total_espera = llamadas_entrantes['total_call_duration'].sum() -␣
 ↪llamadas_entrantes['call_duration'].sum()

# Calcular el número total de llamadas entrantes
numero_llamadas_entrantes = llamadas_entrantes['calls_count'].sum()

# Calcular el tiempo de espera promedio
tiempo_espera_promedio = tiempo_total_espera / numero_llamadas_entrantes

print("El tiempo de espera promedio para las llamadas entrantes es:",␣
 ↪tiempo_espera_promedio, "segundos")
```

```
El tiempo de espera promedio para las llamadas entrantes es: 12.739733972839575
segundos
```

[23]:
```
# Filtrar las llamadas entrantes
llamadas_entrantes = df_llamadas_2[df_llamadas_2['direction'] == 'in']

# Calcular el tiempo de espera para cada llamada
llamadas_entrantes['tiempo_espera'] = llamadas_entrantes['total_call_duration']␣
 ↪- llamadas_entrantes['call_duration']

# Grafico
plt.figure(figsize=(8,6))
sns.boxplot(x=llamadas_entrantes['tiempo_espera'])
plt.title('Distribución del tiempo de espera de las llamadas entrantes')
plt.xlabel('Tiempo de espera (segundos)')
plt.show()
```

```
/tmp/ipykernel_33/3623610701.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  llamadas_entrantes['tiempo_espera'] =
llamadas_entrantes['total_call_duration'] - llamadas_entrantes['call_duration']
```



La distribución del tiempo de espera y su promedio, muestra que la mayoría de las llamadas son atendidas casi inmediatamente. No obstante, la presencia de valores atípicos indica que un grupo de clientes experimenta esperas significativamente más largas. Es necesario investigar las causas de estos tiempos de espera excesivos para optimizar el servicio.

Hipotesis 2: Existe una diferencia significativa en el tiempo de espera promedio entre los operadores "lentos" y los "rápidos"

[24]:
```
percentil_90_tiempo_espera = np.percentile(llamadas_entrantes['tiempo_espera'],␣
↪90)
percentil_90_tiempo_espera
```

[24]:
```
235.0
```

[25]:
```
percentil_90 = 235

# Crear una nueva columna para clasificar a los operadores lentos
```

```
llamadas_entrantes['operadores_lentos'] = llamadas_entrantes['tiempo_espera'] >␣
 ↪percentil_90

llamadas_entrantes.groupby('operadores_lentos')['tiempo_espera'].describe()
```

```
/tmp/ipykernel_33/3583260547.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  llamadas_entrantes['operadores_lentos'] = llamadas_entrantes['tiempo_espera']
> percentil_90
```

[25]:

operadores_lentos	count	mean	std	min	25%	50%	75%	max
False	19334.0	49.029740	51.424383	0.0	12.0	30.0	67.00	235.0
True	2146.0	622.719944	551.832015	236.0	306.0	415.5	703.25	4476.0

El análisis revela que el 10% de los operadores superan el umbral de 235 segundos en tiempo de espera promedio, lo que los clasifica como "lentos". Estos operadores presentan un tiempo de espera promedio casi 13 veces mayor que el resto del grupo.

[26]:
```
# Crear una nueva columna para clasificar a los operadores rapidos
llamadas_entrantes['operadores_rapidos'] = llamadas_entrantes['tiempo_espera']␣
 ↪<= percentil_90

# Extraer los tiempos de espera para cada grupo
tiempos_lentos =␣
 ↪llamadas_entrantes[llamadas_entrantes['operadores_lentos']]['tiempo_espera']
tiempos_rapidos =␣
 ↪llamadas_entrantes[llamadas_entrantes['operadores_rapidos']]['tiempo_espera']

# Verificar normalidad para operadores lentos
stat, p = stats.shapiro(tiempos_lentos)
print('Operadores lentos:')
print('Estadístico de Shapiro-Wilk:', stat)
print('p-valor:', p)
if p > 0.05:
    print('Los datos se distribuyen aproximadamente de forma normal')
else:
    print('Los datos no se distribuyen de forma normal')
```

```
print('------------------------------------------------------------')

# Verificar normalidad para operadores rápidos
stat, p = stats.shapiro(tiempos_rapidos)
print('Operadores rápidos:')
print('Estadístico de Shapiro-Wilk:', stat)
print('p-valor:', p)
if p > 0.05:
    print('Los datos se distribuyen aproximadamente de forma normal')
else:
    print('Los datos no se distribuyen de forma normal')
```

```
Operadores lentos:
Estadístico de Shapiro-Wilk: 0.6568282246589661
p-valor: 0.0
Los datos no se distribuyen de forma normal
------------------------------------------------------------
Operadores rápidos:
Estadístico de Shapiro-Wilk: 0.8137766718864441
p-valor: 0.0
Los datos no se distribuyen de forma normal
```

```
/tmp/ipykernel_33/1183467540.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  llamadas_entrantes['operadores_rapidos'] = llamadas_entrantes['tiempo_espera']
<= percentil_90
/opt/conda/envs/python3/lib/python3.9/site-
packages/scipy/stats/_morestats.py:1816: UserWarning: p-value may not be
accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

[27]:
```
# Prueba de Mann-Whitney U
U1, p = mannwhitneyu(tiempos_lentos, tiempos_rapidos)

print('Estadístico U:', U1)
print('p-valor:', p)

if p < 0.05:
    print("Existe una diferencia significativa en los tiempos de espera entre␣
 ↪los dos grupos.")
else:
    print("No se encontró una diferencia significativa.")
```

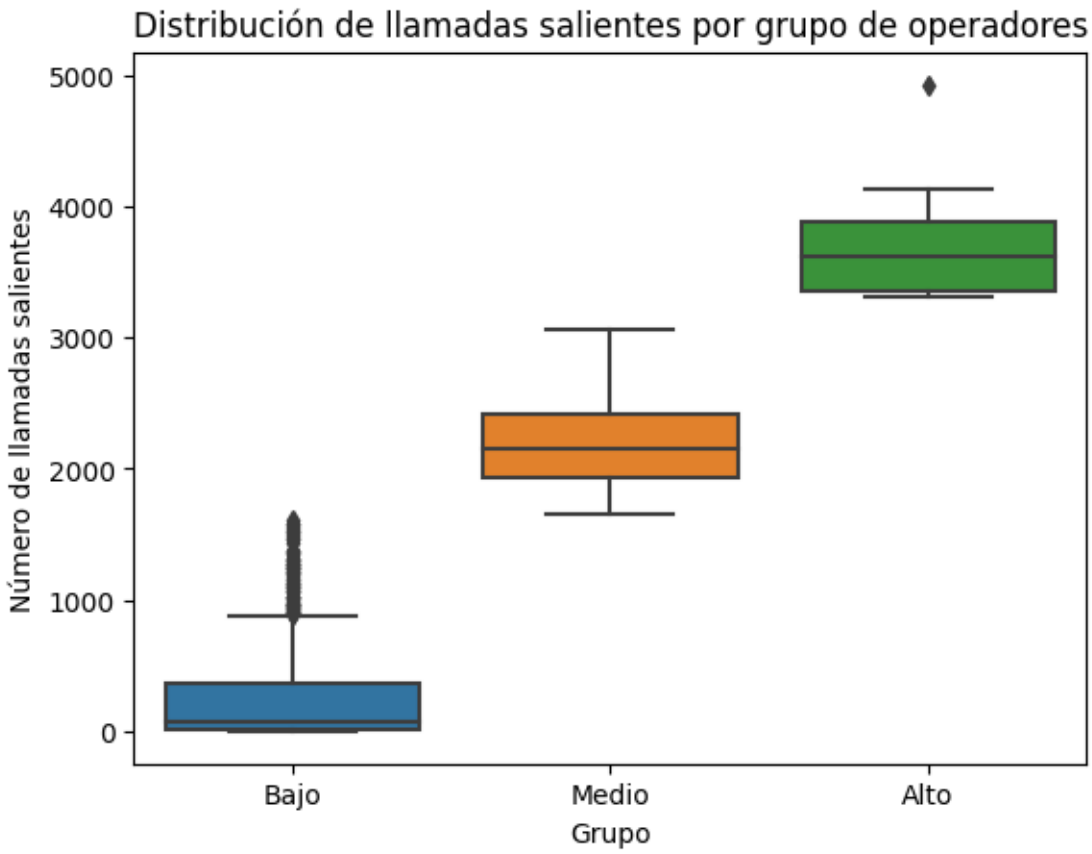
```
Estadístico U: 41490764.0
p-valor: 0.0
Existe una diferencia significativa en los tiempos de espera entre los dos
grupos.
```

Los resultados obtenidos muestran de manera contundente que existe una diferencia significativa en el desempeño de los operadores, evidenciada por los tiempos de espera. Los operadores clasificados como 'lentos' presentaron un rendimiento significativamente inferior ($p < 0.05$) en comparación con sus colegas.

Rechazamos la hipótesis nula (Donde NO hay una diferencia significativa). Esto significa que podemos concluir con un alto grado de confianza que sí existe una diferencia significativa en los tiempos de espera entre los dos grupos de operadores.

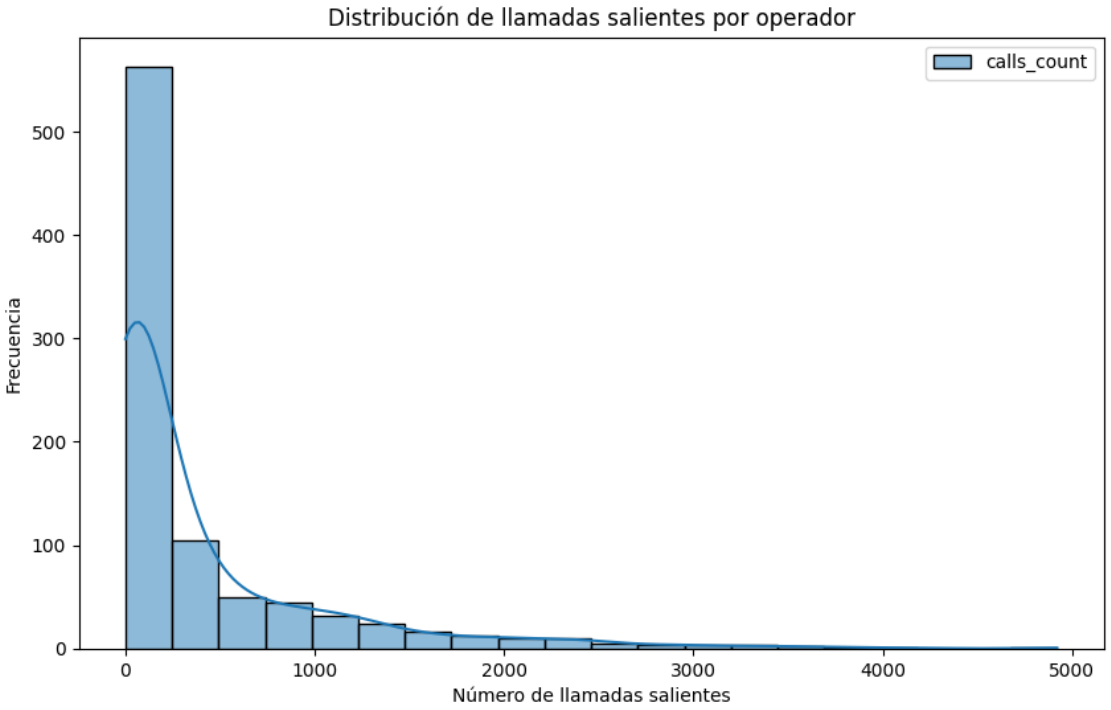
1.3.3 Número de llamadas salientes por operador

[28]:
```
# Filtrar las llamadas salientes
llamadas_salientes = df_llamadas_2[df_llamadas_2['direction'] == 'out']

# Agrupar por operador y contar las llamadas
llamadas_por_operador = llamadas_salientes.
 ↪groupby('operator_id')['calls_count'].sum().reset_index()

# Crear categorías basadas en cuantiles
labels = ['Bajo', 'Medio', 'Alto']
llamadas_por_operador['grupo'] = pd.cut(llamadas_por_operador['calls_count'],␣
 ↪bins=3, labels=labels)

# Crear un boxplot para comparar la distribución por grupos
sns.boxplot(x='grupo', y='calls_count', data=llamadas_por_operador)
plt.title('Distribución de llamadas salientes por grupo de operadores')
plt.xlabel('Grupo')
plt.ylabel('Número de llamadas salientes')
plt.show()
```



Los resultados obtenidos indican una desigual distribución de la carga de trabajo entre los operadores. La alta concentración de llamadas en el grupo "Alto" sugiere la necesidad de redistribuir las tareas o proporcionar recursos adicionales a estos operadores. Además, el análisis del outlier en este grupo podría revelar oportunidades de optimización o áreas donde se requiere atención especial.

[29]:
```
# Histograma
plt.figure(figsize=(10, 6))
sns.histplot(data=llamadas_por_operador, bins=20, kde=True)
plt.title('Distribución de llamadas salientes por operador')
plt.xlabel('Número de llamadas salientes')
plt.ylabel('Frecuencia')
plt.show()
```



[30]:
```
llamadas_por_operador.mean()
```

[30]:
```
calls_count    403.420159
dtype: float64
```

[31]:
```
percentil_25 = np.percentile(llamadas_por_operador['calls_count'], 25)
print("El percentil 25 de llamadas por operador es:", percentil_25)
```

```
El percentil 25 de llamadas por operador es: 12.5
```

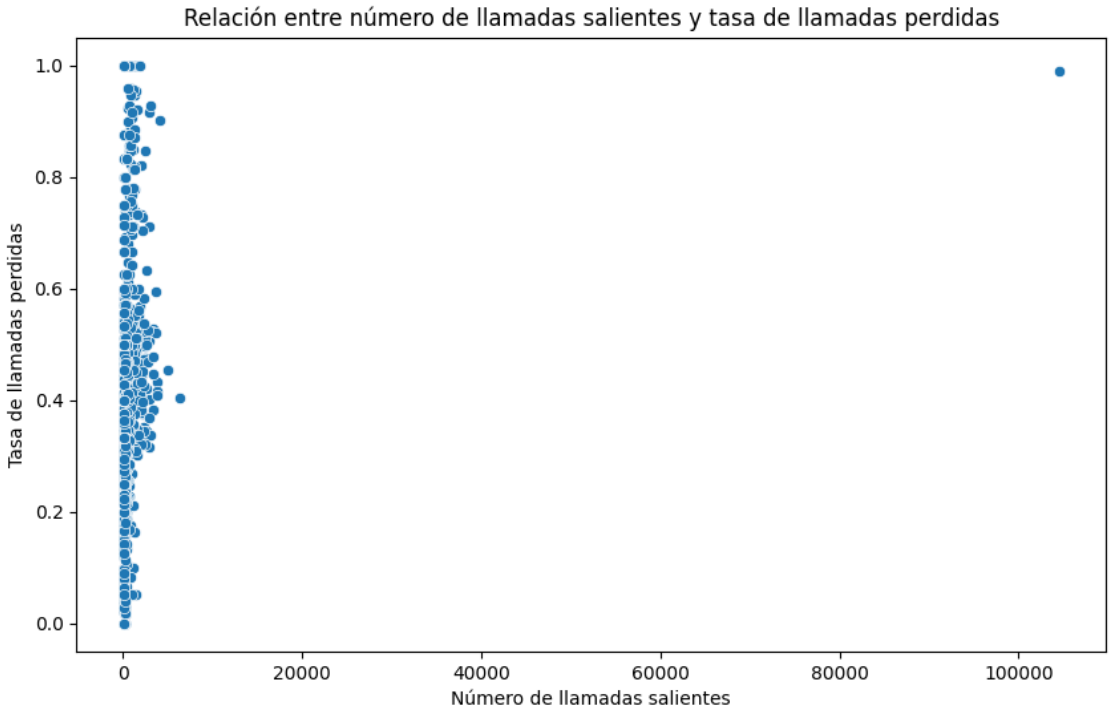
[32]:
```
# Crear una serie con los operadores por debajo del percentil 25
operadores_bajo_percentil =␣
 ↪llamadas_por_operador[llamadas_por_operador['calls_count'] <␣
 ↪percentil_25]['operator_id']

print("Operadores por debajo del percentil 25:")
print(operadores_bajo_percentil)
```

```
Operadores por debajo del percentil 25:
8      882478.0
13     883018.0
14     883898.0
17     884402.0
18     884406.0
```

```
        …
874    970258.0
875    970484.0
876    970486.0
877    972408.0
881    973120.0
Name: operator_id, Length: 221, dtype: object
```

Existe una gran disparidad en el número de llamadas salientes realizadas por cada operador. donde el operador que mas llamadas atiende, tiene un total de 4.921 de llamadas. Un pequeño grupo de operadores concentra un volumen significativamente mayor de llamadas en comparación con el resto del equipo. Hay un número considerable de operadores (221) que realizan muy pocas llamadas salientes, por debajo de 12 llamadas, incluso algunos solo una. Esto podría indicar problemas de asignación de tareas, falta de capacitación o factores externos que afectan su productividad. Posible necesidad de reasignación, la concentración de llamadas en pocos operadores podría indicar una sobrecarga de trabajo para ellos y una posible subutilización de otros. Podría ser necesario reasignar las tareas para distribuir la carga de trabajo de manera más equitativa.

[33]:
```
# Calcular la tasa de llamadas perdidas por operador
grouped_data = df_llamadas_2.groupby('operator_id').
 ↪agg(calls_count=('calls_count', 'sum'),
                                                    ␣
 ↪missed_call_rate=('is_missed_call', 'mean'))

# Grafico
plt.figure(figsize=(10, 6))
sns.scatterplot(x='calls_count', y='missed_call_rate', data=grouped_data)
plt.title('Relación entre número de llamadas salientes y tasa de llamadas␣
 ↪perdidas')
plt.xlabel('Número de llamadas salientes')
plt.ylabel('Tasa de llamadas perdidas')
plt.show()
```



No se observa una relación clara entre el número de llamadas salientes y la tasa de llamadas perdidas. Otros factores, como la complejidad de las llamadas o la disponibilidad de los agentes, podrían estar influyendo en la tasa de pérdidas.

1.4 Conclusion y recomendaciones

El análisis exhaustivo de los datos de llamadas ha revelado una heterogeneidad significativa en el desempeño de los operadores de CallMeMaybe. Se identificaron brechas significativas en términos de volumen de llamadas, tiempos de espera, tasas de llamadas perdidas y eficiencia en la resolución de incidentes.

Los resultados evidencian que un porcentaje considerable de operadores no cumple con los estándares establecidos, lo que impacta directamente en la satisfacción del cliente y la reputación de la empresa. Las causas de este bajo desempeño son multifactoriales y abarcan desde la falta de capacitación y experiencia hasta problemas en la asignación de tareas y la gestión de recursos.

*Recomendaciones:

1. Mejora de las condiciones laborales y el trato: Reconocimiento y valoración: Implementar un programa de reconocimiento que valore las contribuciones de los operadores y celebre sus logros. Esto puede incluir bonos, días libres adicionales o simplemente un reconocimiento público. Clima laboral positivo: Fomentar un ambiente de trabajo colaborativo y respetuoso, donde los operadores se sientan valorados y apoyados. Esto puede incluir actividades de team building, canales de comunicación abiertos y la resolución de conflictos de manera justa y transparente. Flexibilidad: Evaluar la posibilidad de ofrecer horarios flexibles o jornadas reducidas, siempre que sea compatible con las necesidades del negocio. Esto puede ayudar

a mejorar la conciliación de la vida laboral y personal y aumentar la satisfacción de los empleados. Programas de bienestar: Implementar programas de bienestar que promuevan la salud física y mental de los empleados, como1 clases de yoga, programas de nutrición o asistencia psicológica.

2. Revisión de la remuneración: Evaluación de salarios: Realizar un estudio de mercado para determinar si los salarios actuales son competitivos y justos en comparación con otras empresas del sector. Estructura salarial transparente: Establecer una estructura salarial clara y transparente, basada en el desempeño y la experiencia. Incentivos por desempeño: Implementar un sistema de incentivos que recompense a los operadores por alcanzar metas y objetivos establecidos. Beneficios adicionales: Ofrecer beneficios adicionales como seguro médico, seguro de vida o descuentos en productos o servicios de la empresa.

3. Fortalecimiento de la capacitación y el desarrollo: Capacitación continua: Implementar un programa de capacitación continua que aborde las necesidades específicas de cada operador y las últimas tendencias del sector. Reconocimiento de certificaciones: Reconocer y recompensar a los operadores que obtengan certificaciones profesionales relacionadas con su trabajo. Mentoría: Implementar un programa de mentoría que permita a los operadores más experimentados compartir sus conocimientos con los nuevos empleados.

4. Comunicación efectiva: Canales de comunicación abiertos: Establecer canales de comunicación bidireccionales para que los operadores puedan expresar sus inquietudes y sugerencias. Encuestas de clima laboral: Realizar encuestas de clima laboral de forma regular para identificar áreas de mejora y tomar medidas correctivas.

Al implementar estas recomendaciones, CallMeMaybe podrá mejorar significativamente la calidad de su servicio, aumentar la satisfacción del cliente y fortalecer su posición en el mercado. Es fundamental que la empresa se comprometa con un proceso de mejora continua y que invierta en el desarrollo de sus operadores para garantizar el éxito a largo plazo

1.5 Link para presentacion y dashboard

https://drive.google.com/drive/folders/1WpDE0LrWlR1Q3MQYN6jtaozbe4EBrrjR?usp=sharing