

Analisis de abandono de clientes

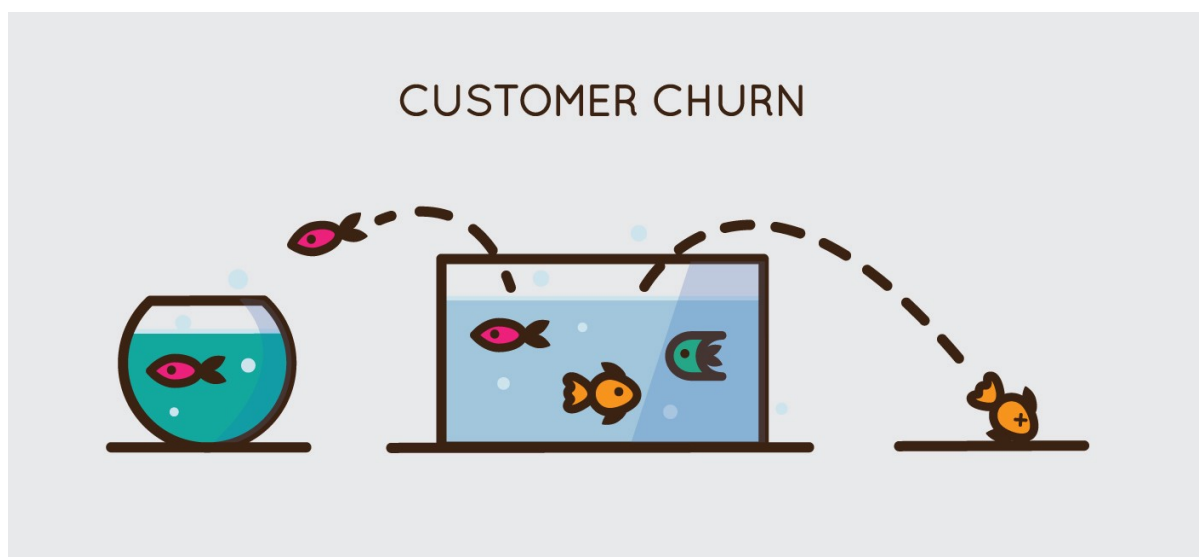
Descripcion del proyecto

Definición del objetivo: El objetivo principal es analizar el comportamiento de los clientes de una compañía de telecomunicaciones y predecir la probabilidad de cancelación (churn) de un cliente, basándose en variables demográficas, de uso del servicio y de satisfacción. Al predecir con precisión el churn, la empresa puede tomar medidas proactivas para retener a los clientes y mejorar sus servicios.

Contexto comercial: La compañía de telecomunicaciones enfrenta una creciente competencia en el mercado, lo que provoca una alta tasa de cancelación de sus servicios. Retener a los clientes existentes es fundamental para mantener y mejorar la rentabilidad y la cuota de mercado. La identificación de factores clave que influyen en la cancelación permitirá a la empresa diseñar estrategias de retención de clientes más efectivas y personalizadas.

Problema comercial: El problema comercial consiste en identificar y comprender los factores que influyen en la decisión de un cliente de cancelar los servicios de la compañía de telecomunicaciones. El churn de clientes puede generar pérdidas financieras significativas y afectar la reputación de la empresa. Por lo tanto, es esencial predecir y abordar las causas del churn antes de que los clientes decidan cancelar.

Contexto analítico: El enfoque analítico se basa en el análisis exploratorio de datos y en la aplicación de técnicas de modelado predictivo. Primero, se realiza un análisis exploratorio de datos para comprender la estructura y las características del conjunto de datos. Este análisis incluye la revisión de estadísticas descriptivas, la identificación de variables categóricas y continuas, y la detección de valores atípicos y faltantes. Se investigan las relaciones entre las variables demográficas, de uso del servicio y de satisfacción con la tasa de cancelación (churn). Para ello, se pueden utilizar gráficos de barras, diagramas de caja y gráficos de dispersión, dependiendo del tipo de variables que se estén analizando.



Descripcion del Dataset

El data set se basa en un conjunto de datos de IBM Telco Customer Churn.

La compañía JB Link, una pequeña empresa de telecomunicaciones en California, enfrenta una alta tasa de churn de clientes lo que significa que muchos clientes están cancelando sus servicios o dejando de usarlos y ha creado un equipo de trabajo, que incluye un equipo de ciencia de datos, para desarrollar una estrategia de retención de clientes.

El equipo de ciencia de datos se encargará de obtener información de los datos, desarrollar un modelo de aprendizaje automático para predecir la churn de clientes y prescribir acciones personalizadas para retener a cada uno de esos clientes. Los ejecutivos esperan que el proyecto ahorre mucho dinero a la compañía y la haga crecer nuevamente.

Importacion de librerias utilizadas

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.preprocessing import OneHotEncoder

import plotly.express as px
import plotly.figure_factory as ff
import plotly.graph_objects as go

from lifelines import KaplanMeierFitter
```

Importacion del data set

```
In [2]: filename = 'DataSet-Troconiz-E1.csv'
url = "DataSet-Troconiz-E1.csv"
dataset = pd.read_csv(url)
```

Muestreo de datos de todas las columnas:

```
In [3]: pd.set_option('display.max_columns', 50)
dataset.head()
```

Out[3]:

	Customer ID	Referred a Friend	Number of Referrals	Tenure in Months	Offer	Phone Service	Avg Monthly Long Distance Charges	Multiple Lines	Internet Service	Internet Type
0	8779-QRDMV	No	0	1	None	No	0.00	No	Yes	Fiber Optic
1	7495-OOKFY	Yes	1	8	Offer E	Yes	48.85	Yes	Yes	Cable
2	1658-BYGOY	No	0	18	Offer D	Yes	11.33	Yes	Yes	Fiber Optic
3	4598-XLKNJ	Yes	1	25	Offer C	Yes	19.76	No	Yes	Fiber Optic
4	4846-WHAFZ	Yes	1	37	Offer C	Yes	6.33	Yes	Yes	Cable

Informacion referente a las columnas

```
In [4]: dataset['Customer ID'].unique
```

```
Out[4]: <bound method Series.unique of 0      8779-QRDMV
1      7495-OOKFY
2      1658-BYGOY
3      4598-XLKNJ
4      4846-WHAFZ
...
7038    2569-WGERO
7039    6840-RESVB
7040    2234-XADUH
7041    4801-JZAZL
7042    3186-AJIEK
Name: Customer ID, Length: 7043, dtype: object>
```

In [5]: dataset.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 46 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   Customer ID                             7043 non-null   object
1   Referred a Friend                       7043 non-null   object
2   Number of Referrals                     7043 non-null   int64
3   Tenure in Months                       7043 non-null   int64
4   Offer                                  7043 non-null   object
5   Phone Service                          7043 non-null   object
6   Avg Monthly Long Distance Charges      7043 non-null   float64
7   Multiple Lines                         7043 non-null   object
8   Internet Service                       7043 non-null   object
9   Internet Type                          7043 non-null   object
10  Avg Monthly GB Download                 7043 non-null   int64
11  Online Security                        7043 non-null   object
12  Online Backup                          7043 non-null   object
13  Device Protection Plan                  7043 non-null   object
14  Premium Tech Support                   7043 non-null   object
15  Streaming TV                           7043 non-null   object
16  Streaming Movies                       7043 non-null   object
17  Streaming Music                        7043 non-null   object
18  Unlimited Data                         7043 non-null   object
19  Contract                               7043 non-null   object
20  Paperless Billing                       7043 non-null   object
21  Payment Method                         7043 non-null   object
22  Monthly Charge                         7043 non-null   float64
23  Total Regular Charges                   7043 non-null   float64
24  Total Refunds                          7043 non-null   float64
25  Total Extra Data Charges                7043 non-null   float64
26  Total Long Distance Charges             7043 non-null   float64
27  Gender                                 7043 non-null   object
28  Age                                    7043 non-null   int64
29  Under 30                              7043 non-null   object
30  Senior Citizen                         7043 non-null   object
31  Married                               7043 non-null   object
32  Dependents                            7043 non-null   object
33  Number of Dependents                   7043 non-null   int64
34  City                                  7043 non-null   object
35  Zip Code                              7043 non-null   int64
36  Latitude                              7043 non-null   float64
37  Longitude                             7043 non-null   float64
38  Population                            7043 non-null   int64
39  Churn Value                            7043 non-null   int64
40  CLTV                                  7043 non-null   int64
41  Churn Category                         1869 non-null   object
42  Churn Reason                           1869 non-null   object
43  Total Customer Svc Requests            7043 non-null   int64
44  Product/Service Issues Reported        7043 non-null   int64
45  Customer Satisfaction                  1834 non-null   float64
dtypes: float64(9), int64(11), object(26)
memory usage: 2.5+ MB
```

Descripcion de las columnas:

0 ID del Cliente: Identificación única del cliente.

1 Recomendó a un Amigo: Si el cliente ha referido a amigos al servicio.

2 Número de Referidos: La cantidad de personas que el cliente ha referido al servicio.

3 Antigüedad en Meses: La duración de la relación del cliente con el servicio, en meses.

4 Oferta: Detalles sobre cualquier oferta especial que se le haya proporcionado al cliente.

5 Servicio de Teléfono: Si el cliente ha contratado un servicio telefónico.

6 Cargos Promedio Mensuales de Larga Distancia: Monto promedio que el cliente paga por llamadas de larga distancia.

7 Líneas Múltiples: Si el cliente tiene múltiples líneas telefónicas.

8 Servicio de Internet: Si el cliente ha contratado un servicio de internet.

9 Tipo de Internet: El tipo de conexión a Internet del cliente (por ejemplo, DSL, fibra, etc.)

10 Descarga Promedio Mensual en GB: La cantidad promedio de datos que el cliente descarga cada mes.

11 Seguridad en Línea: Si el cliente tiene un servicio de seguridad en línea.

12 Respaldo en Línea: Si el cliente tiene un servicio de respaldo en línea.

13 Plan de Protección de Dispositivos: Si el cliente tiene un plan de protección para sus dispositivos.

14 Soporte Técnico Premium: Si el cliente tiene un servicio de soporte técnico premium.

15 TV por Streaming: Si el cliente usa servicios de streaming de TV.

16 Películas por Streaming: Si el cliente usa servicios de streaming de películas.

17 Música por Streaming: Si el cliente usa servicios de streaming de música.

18 Datos Ilimitados: Si el cliente tiene un plan de datos ilimitados.

19 Contrato: El tipo de contrato del cliente con el servicio.

20 Facturación sin Papel: Si el cliente utiliza la opción de facturación sin papel.

21 Método de Pago: Método de pago utilizado por el cliente.

22 Cargo Mensual: El costo mensual de los servicios para el cliente.

23 Total de Cargos Regulares: El total de los cargos regulares que el cliente ha pagado.

24 Total de Reembolsos: La cantidad total que se ha reembolsado al cliente.

25 Total de Cargos por Datos Extra: El total que el cliente ha pagado por los datos extra.

26 Total de Cargos de Larga Distancia: El total que el cliente ha pagado por las llamadas de larga distancia.

27 Género: El género del cliente.

28 Edad: La edad del cliente.

29 Menor de 30: Si el cliente es menor de 30 años de edad.

30 Ciudadano Mayor: Si el cliente es considerado un ciudadano mayor.

31 Casado: Si el cliente está casado.

32 Dependientes: Si el cliente tiene dependientes.

33 Número de Dependientes: El número de dependientes que tiene el cliente.

34 Ciudad: La ciudad donde reside el cliente.

35 Código Postal: El código postal de la residencia del cliente.

36 Latitud: La latitud de la residencia del cliente.

37 Longitud: La longitud de la residencia del cliente.

38 Población: La población de la ciudad donde reside el cliente.

39 Valor de Churn: Si el cliente ha abandonado el servicio (1) o no (0).

40 CLTV: El valor de vida del cliente (Customer Lifetime Value).
 41 Categoría de Churn: La razón general por la que el cliente abandonó el servicio.
 42 Motivo de Churn: El motivo específico por el que el cliente abandonó el servicio.
 43 Total de Solicitudes al Servicio al Cliente: El número total de veces que el cliente ha contactado al servicio al cliente.
 44 Problemas de Producto/Servicio Reportados: El número total de problemas de producto o servicio que el cliente ha reportado.
 45 Satisfacción del Cliente: El nivel de satisfacción del cliente con el servicio, en una escala del 1 al 5.

Resumen Estadístico del Conjunto de Datos

El método `.describe()` de Pandas proporciona un resumen estadístico de alto nivel de las columnas numéricas en un DataFrame. Este resumen incluye la cuenta (número de elementos no nulos), la media, la desviación estándar, los valores mínimos y máximos y los percentiles de las columnas numéricas.

In [6]: `dataset.describe()`

Out[6]:

	Number of Referrals	Tenure in Months	Avg Monthly Long Distance Charges	Avg Monthly GB Download	Monthly Charge	Total Regular Charges	Tot Refunc
count	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000
mean	1.951867	32.386767	22.958954	21.110890	65.538800	2280.381264	1.96218
std	3.001199	24.542061	15.448113	20.948471	30.606805	2266.220462	7.90267
min	0.000000	1.000000	0.000000	0.000000	18.250000	18.800000	0.000000
25%	0.000000	9.000000	9.210000	3.000000	35.890000	400.150000	0.000000
50%	0.000000	29.000000	22.890000	17.000000	71.968000	1394.550000	0.000000
75%	3.000000	55.000000	36.395000	28.000000	90.650000	3786.600000	0.000000
max	11.000000	72.000000	49.990000	94.000000	123.084000	8684.800000	49.790000

Estimador Kaplan-Meier para el Análisis de Churn

En este cuaderno, realizaremos un análisis de Churn (tasa de abandono de clientes) utilizando el estimador Kaplan-Meier. La 'duración' representa el tiempo en meses que un cliente ha estado usando el servicio, y el 'valor de churn' indica si el cliente ha dejado el servicio. Utilizaremos la biblioteca `lifelines` en Python para llevar a cabo este análisis.

In [7]:

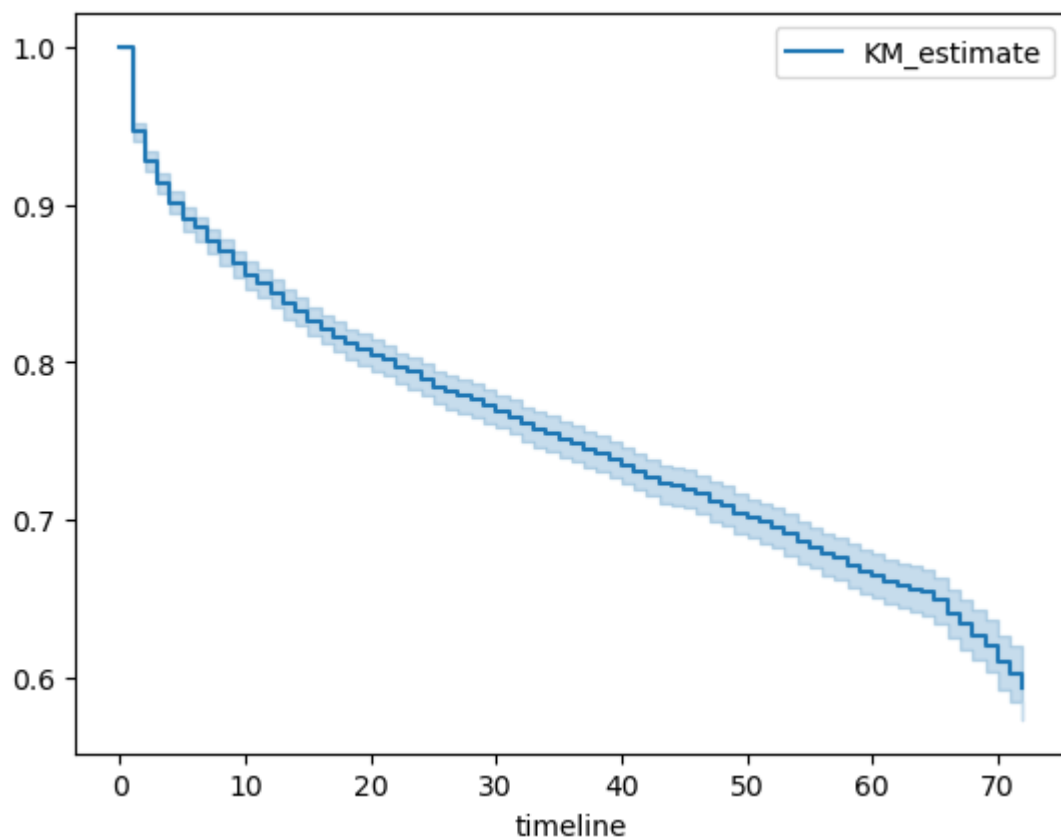
```
# Asegúrate de que las columnas "Churn Value" y "tenure in months" estén en el
dataset['Churn Value'] = pd.to_numeric(dataset['Churn Value'], errors='coerce')
dataset['tenure in months'] = pd.to_numeric(dataset['Tenure in Months'], errors='coerce')

# Crear una instancia de KaplanMeierFitter
kmf = KaplanMeierFitter()

# Ajustar los datos al modelo
kmf.fit(dataset['Tenure in Months'], dataset['Churn Value'])

# Crear un gráfico de la estimación de Kaplan-Meier
kmf.plot_survival_function()
```

Out[7]: <AxesSubplot:xlabel='timeline'>



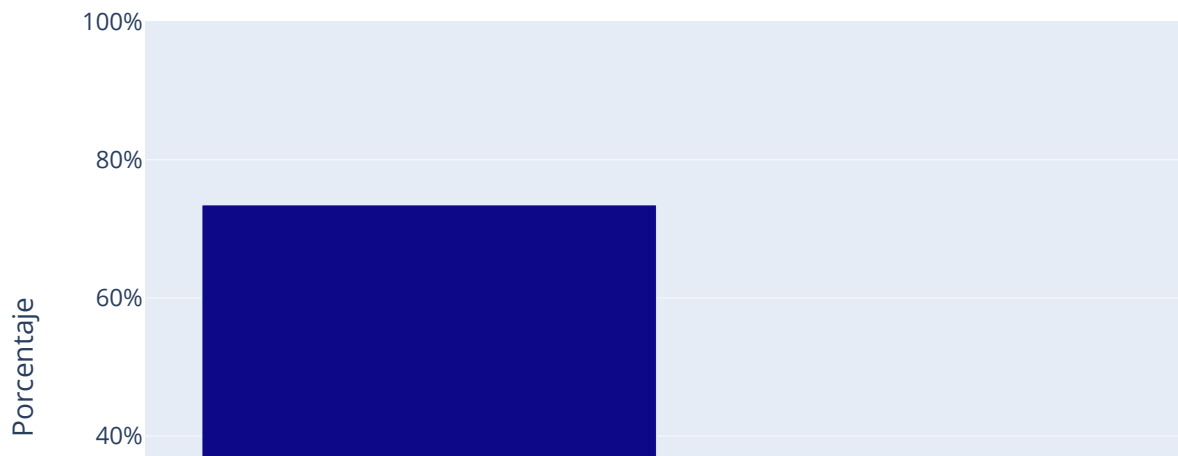
Visualizacion del abandono

El gráfico 'Distribución de la Tasa de Cancelación' muestra cuántos clientes han abandonado nuestros servicios (valor 1) en comparación con los que permanecen (valor 0). Esta representación nos proporciona una clara imagen de nuestro rendimiento en la retención de clientes, lo que nos permite diseñar estrategias para mejorar nuestra tasa de retención y reducir el abandono.

```
In [8]: # Calcular la frecuencia relativa
freq_rel = dataset['Churn Value'].value_counts(normalize=True)
# Convertir a DataFrame y resetear el índice
df_freq_rel = freq_rel.to_frame().reset_index()
df_freq_rel.columns = ['Churn Value', 'Frecuencia Relativa']
fig = px.bar(df_freq_rel,
             x='Churn Value',
             y='Frecuencia Relativa',
             title='Distribución de la Tasa de Cancelación',
             labels={'Churn Value': 'Valor de Churn, 0 no abandono, 1 abandono',
                    'Frecuencia Relativa': 'Porcentaje'},
             color='Churn Value')

fig.update_yaxes(tickformat=".0%", range=[0, 1]) # Formato de porcentaje en e
fig.update_xaxes(tickvals=[0, 1])
fig.show()
```

Distribución de la Tasa de Cancelación



Nuestros datos muestran que de un total de 7043 clientes, 73 % (5174) han elegido continuar con nuestros servicios, mientras que 26% (1869) han decidido abandonarnos. Aunque la mayoría de nuestros clientes siguen confiando en nosotros, no podemos ignorar que más de

una cuarta parte ha optado por alternativas diferentes. Este nivel de abandono resalta la necesidad de mejorar nuestras estrategias de retención de clientes para asegurarnos de mantener nuestra base de clientes y seguir siendo competitivos en el mercado

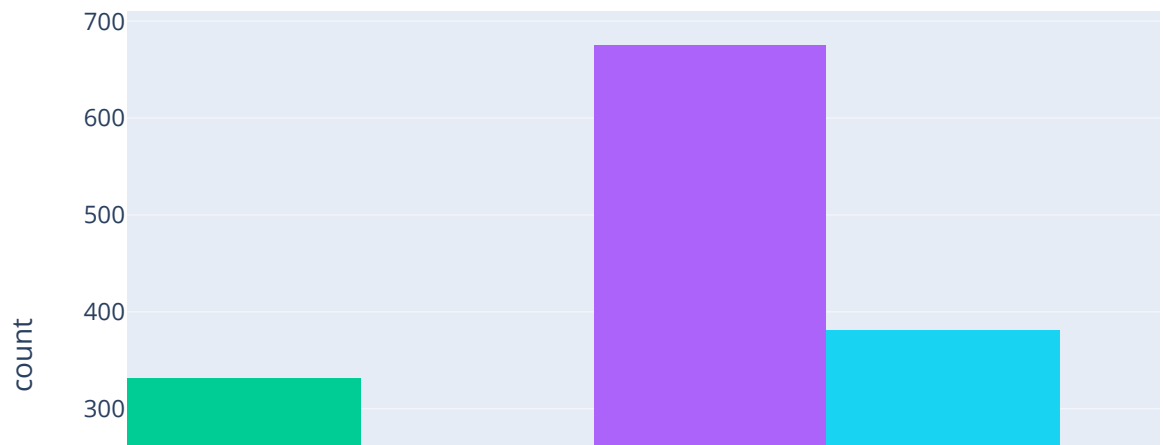
Visualizacion de la satisfaccion del cliente



Nuestro gráfico 'Distribución de la Satisfacción del Cliente' representa cómo nuestros clientes califican su experiencia con nuestros servicios, en una escala de 1 a 5. Cada barra del gráfico indica la cantidad de clientes que han dado una puntuación específica, lo que nos permite identificar rápidamente cuál es la satisfacción promedio y ver si la mayoría de nuestros clientes están contentos con lo que ofrecemos. Esta visión nos permite entender mejor nuestras fortalezas y las áreas donde necesitamos mejorar para continuar ofreciendo un servicio de alta calidad a nuestros clientes.

```
In [9]: fig = px.histogram(dataset,
                        x="Customer Satisfaction",
                        nbins=5, # Número de barras.
                        title='Distribución de la Satisfacción del Cliente',
                        labels={'Customer Satisfaction': 'Satisfacción del Cliente, '},
                        color='Customer Satisfaction') # Color de las barras. En es
fig.show()
```

Distribución de la Satisfacción del Cliente



```
In [10]: conteo_satisfaccion = dataset['Customer Satisfaction'].value_counts().sort_index()
conteo_satisfaccion = conteo_satisfaccion.astype(int)
total_encuestas = conteo_satisfaccion.sum()
porcentaje_satisfaccion = round((conteo_satisfaccion / total_encuestas) * 100, 2)
df_satisfaccion = pd.DataFrame({'Conteo': conteo_satisfaccion, 'Porcentaje (%)': porcentaje_satisfaccion})
df_satisfaccion.loc['Total'] = [total_encuestas, 100.00]
df_satisfaccion.loc['Total', 'Conteo'] = df_satisfaccion.loc['Total', 'Conteo']
df_satisfaccion
```

Out[10]:

	Conteo	Porcentaje (%)
1.0	332.0	18.10
2.0	200.0	10.91
3.0	675.0	36.80
4.0	380.0	20.72
5.0	247.0	13.47
Total	1834.0	100.00

Nuestro estudio de la 'Satisfacción del Cliente' se basó en las respuestas de 1834 encuestados. Según los datos recopilados, el nivel de satisfacción más frecuente es 3, representando el 36.80% de las respuestas. Por otro lado, los niveles de satisfacción 1 y 2, que representan experiencias menos satisfactorias, comprenden el 18.10% y 10.91% respectivamente. Los niveles de satisfacción más altos, 4 y 5, representan el 20.72% y 13.47% respectivamente.

Esto sugiere que, aunque la mayoría de nuestros clientes están satisfechos con nuestros servicios (con una calificación de 3 o superior), aún hay un porcentaje significativo que tiene una experiencia menos positiva. Estos hallazgos resaltan la necesidad de continuar nuestros esfuerzos para mejorar la experiencia del cliente y así incrementar el porcentaje de clientes con los más altos niveles de satisfacción.

Resumen

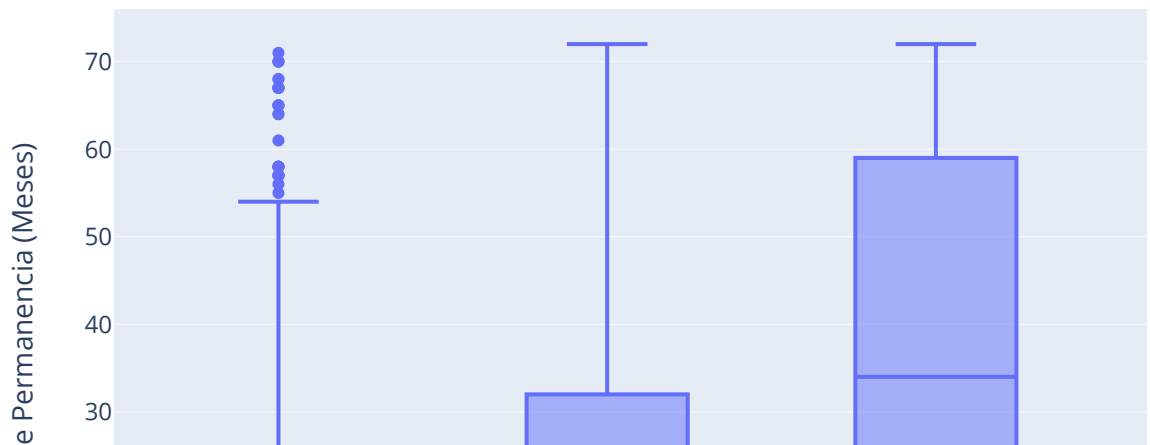
De las 1834 respuestas recopiladas en la encuesta de satisfacción del cliente, notamos que 532 clientes están insatisfechos o poco satisfechos (niveles de satisfacción 1 y 2). Esto representa cerca del 29% de los encuestados. Además, una proporción significativa de los clientes, alrededor del 37%, tiene una satisfacción neutral (nivel 3). Por otro lado, un 34% de los encuestados están bastante o muy satisfechos con nuestros servicios (niveles 4 y 5). A pesar de este último porcentaje positivo, nos preocupa el segmento insatisfecho y neutral. Por lo tanto, vamos a analizar en detalle la relación entre la satisfacción del cliente y la tasa de abandono, buscando mejorar la experiencia del cliente y reducir la tasa de abandono.

¿Existe una relación entre la satisfacción del cliente y la duración de la suscripción?

Hipótesis: Los clientes más satisfechos podrían tener una mayor duración de la suscripción

```
In [11]: fig = px.box(dataset, x='Customer Satisfaction', y='Tenure in Months',  
                    labels={  
                        "Customer Satisfaction": "Satisfacción del Cliente, mínimo  
                        "Tenure in Months": "Tiempo de Permanencia (Meses)"  
                    },  
                    title='Satisfacción del Cliente vs. Tiempo de Permanencia (Boxplot)',  
                    fig.show()
```

Satisfacción del Cliente vs. Tiempo de Permanencia (Boxplot)



En general, se aprecia una relación entre la satisfacción y el tiempo de permanencia, lo que podría indicar que los clientes que permanecen con la empresa por más tiempo tienden a estar más satisfechos con el servicio. Sin embargo, también observamos algunos valores atípicos que merecen una investigación más profunda. Además, la distribución de los datos sugiere que podríamos tener problemas específicos con los clientes más nuevos o menos satisfechos que necesitamos abordar.

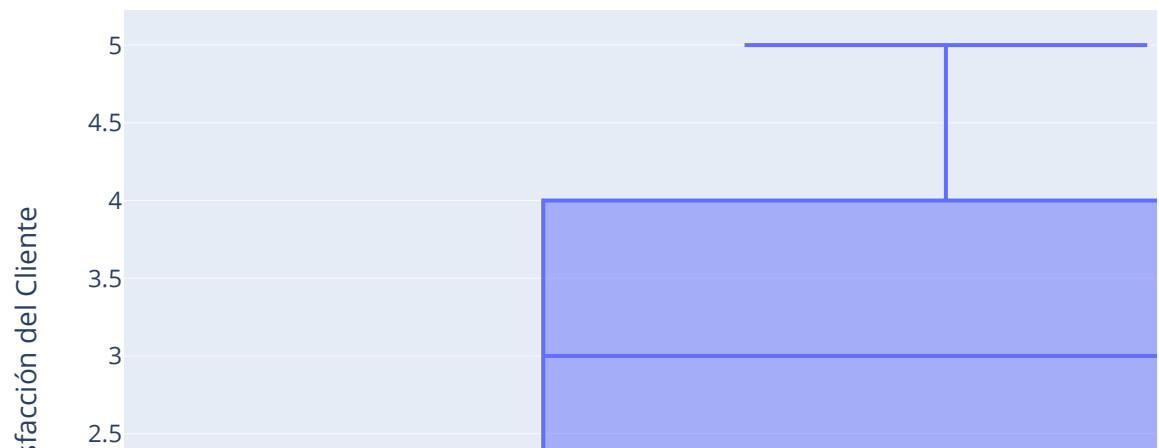
¿Existe una relación entre la tasa de cancelación y la satisfacción del cliente?

Hipótesis: Los clientes menos satisfechos podrían tener una mayor tasa de cancelación debido

```
In [12]: import plotly.express as px

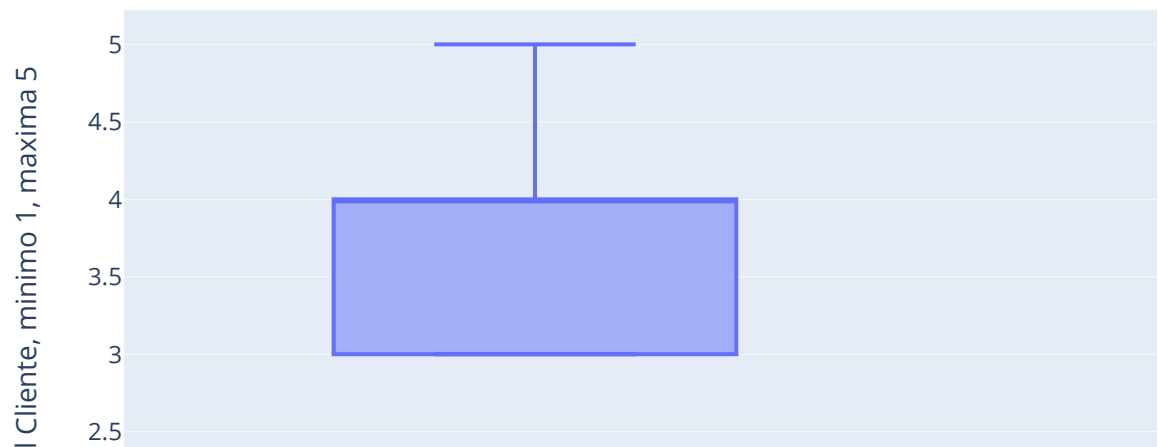
fig = px.box(dataset, y="Customer Satisfaction",
              title='Boxplot de la Satisfacción del Cliente',
              labels={"Customer Satisfaction": "Satisfacción del Cliente"})
fig.show()
```

Boxplot de la Satisfacción del Cliente



```
In [13]: fig = px.box(dataset, x='Churn Value', y='Customer Satisfaction',  
                    labels={  
                        "Churn Value": "Tasa de Cancelación, 0 no abandono, 1 aban  
                        "Customer Satisfaction": "Satisfacción del Cliente, mínimo  
                    },  
                    title='Tasa de Cancelación vs. Satisfacción del Cliente (Boxplot)  
fig.show()
```

Tasa de Cancelación vs. Satisfacción del Cliente (Boxplot)



Los datos visuales nos proporcionan una relación importante entre la satisfacción del cliente y la tasa de abandono. En particular, observamos que los clientes menos satisfechos tienden a abandonar nuestros servicios con mayor frecuencia. Esto subraya la importancia de nuestro enfoque en mejorar la satisfacción del cliente, ya que parece ser un indicador significativo de la lealtad del cliente. Es crucial que desarrollemos estrategias para aumentar la satisfacción del cliente, que podrían incluir mejorar la calidad del servicio, el soporte al cliente, los precios, entre otros factores. Al hacerlo, esperamos reducir la tasa de abandono y mantener a nuestros valiosos clientes a largo plazo.

¿Influye el tiempo de permanencia en la tasa de cancelación?

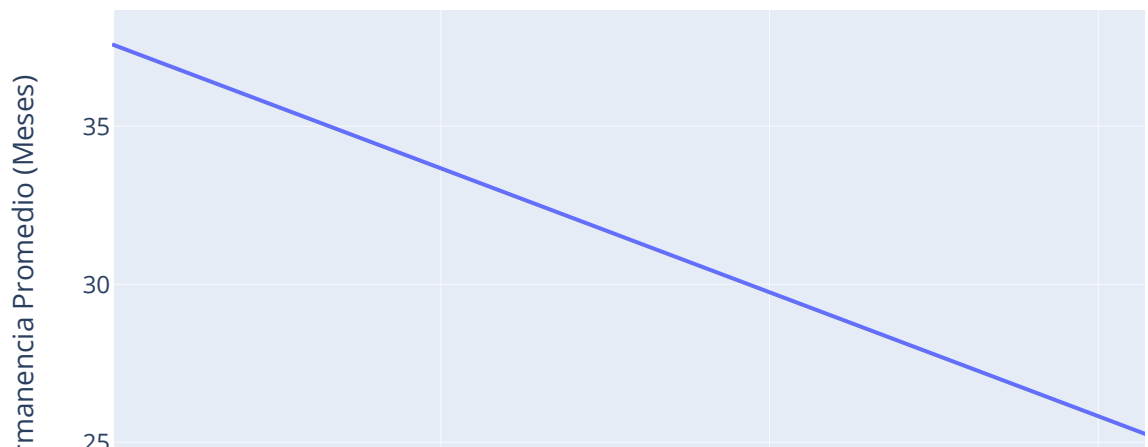
Hipótesis: Los clientes que permanecen con la empresa por menos tiempo podrían tener una mayor tasa de cancelación debido a una menor satisfacción o lealtad hacia la empresa en comparación con clientes de mayor antigüedad.

```
In [14]: mean_tenure = dataset.groupby('Churn Value')['Tenure in Months'].mean().reset_index()

fig = px.line(mean_tenure, x='Churn Value', y='Tenure in Months',
              labels={
                  "Churn Value": "Tasa de Cancelación, 0 no abandono, 1 abandono",
                  "Tenure in Months": "Tiempo de Permanencia Promedio (Meses)"
              },
              title='Tasa de Cancelación vs. Tiempo de Permanencia Promedio')

fig.show()
```

Tasa de Cancelación vs. Tiempo de Permanencia Promedio



Se observa una relación entre la tasa de cancelación y el tiempo de permanencia de los clientes. Los datos indican que hay un menor abandono en los primeros meses de la relación con la empresa. Esto enfatiza la importancia de brindar una experiencia excepcional desde el

inicio para aumentar la retención de clientes a largo plazo. Al centrar nuestros esfuerzos en mejorar la satisfacción y fidelidad durante las etapas críticas del ciclo de vida del cliente, podemos reducir la tasa de cancelación y fortalecer nuestra base de clientes.

Análisis de los Factores que Afectan la Satisfacción del Cliente

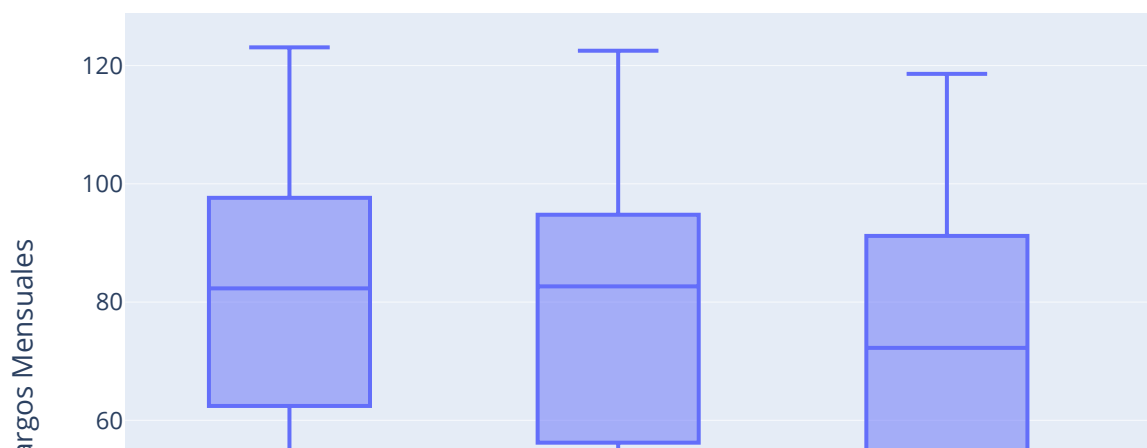
En este estudio, nos proponemos investigar los diferentes factores que pueden influir en la satisfacción del cliente. Nuestro objetivo es identificar las variables clave que tienen un impacto significativo en la satisfacción del cliente y utilizar estos conocimientos para mejorar la calidad de nuestros productos y servicios. A través de un análisis exhaustivo de nuestros datos, buscaremos patrones y correlaciones que nos ayuden a comprender mejor las necesidades y expectativas de nuestros clientes. Además, exploraremos la relación entre la satisfacción del cliente y variables como la calidad del producto, el tiempo de respuesta del servicio al cliente, la facilidad de uso y otros factores relevantes. Con estos hallazgos, podremos tomar medidas concretas para mejorar la satisfacción del cliente y, en última instancia, fortalecer nuestra relación con ellos.

¿Influyen los cargos mensuales en la satisfacción del cliente y la tasa de cancelación?

Hipótesis: Los clientes que pagan cargos mensuales más altos podrían estar menos satisfechos y tener una mayor tasa de cancelación debido a la percepción de un valor insuficiente por su dinero.


```
In [15]: fig = px.box(dataset, x='Customer Satisfaction', y='Monthly Charge',  
                    labels={  
                        "Customer Satisfaction": "Satisfacción del Cliente, mínimo",  
                        "Monthly Charge": "Cargos Mensuales"  
                    },  
                    title='Satisfacción del Cliente vs. Cargos Mensuales (Boxplot)')  
fig.show()
```

Satisfacción del Cliente vs. Cargos Mensuales (Boxplot)

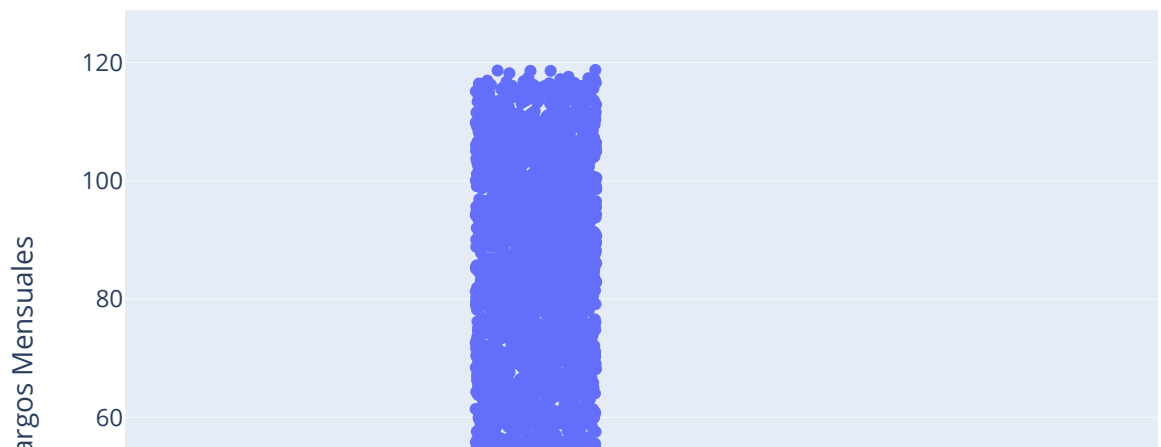


```
In [16]: import plotly.express as px

fig = px.strip(dataset, x='Churn Value', y='Monthly Charge',
               labels={
                   "Churn Value": "Tasa de Cancelación, 0 no abandono, 1 abandono",
                   "Monthly Charge": "Cargos Mensuales"
               },
               title='Tasa de Cancelación vs. Cargos Mensuales (Strip Plot)')

fig.show()
```

Tasa de Cancelación vs. Cargos Mensuales (Strip Plot)



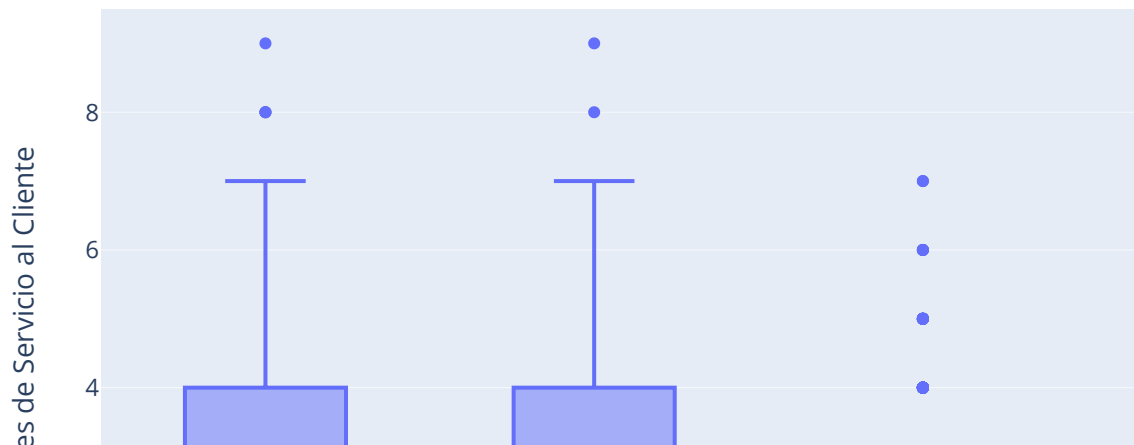
Observamos una correlación evidente en los datos entre cargos mensuales más altos y una disminución en la satisfacción del cliente. Esto indica que a medida que los cargos mensuales aumentan, la satisfacción del cliente tiende a disminuir. Además, notamos que a medida que la satisfacción del cliente disminuye, la tasa de abandono tiende a aumentar. Estos hallazgos sugieren la importancia de considerar cuidadosamente la fijación de precios y la calidad del servicio para mantener la satisfacción del cliente y prevenir el abandono. Es fundamental realizar un análisis más profundo de los factores que influyen en la satisfacción del cliente y utilizar estos conocimientos para tomar medidas concretas y mejorar la experiencia del cliente, tanto en términos de precios competitivos como de calidad del servicio ofrecido.

¿Existe una relación entre el número de solicitudes de servicio al cliente y la satisfacción del cliente?

Hipótesis: Los clientes que realizan más solicitudes de servicio al cliente podrían estar menos satisfechos y tener una mayor tasa de cancelación debido a problemas recurrentes con el servicio o producto.

```
In [17]: fig = px.box(dataset, x='Customer Satisfaction', y='Total Customer Svc Requests',
                    labels={
                        "Customer Satisfaction": "Satisfacción del Cliente, mínimo",
                        "Total Customer Svc Requests": "Solicitudes de Servicio al Cliente",
                    },
                    title='Satisfacción del Cliente vs. Solicitudes de Servicio al Cliente',
                    fig.show()
```

Satisfacción del Cliente vs. Solicitudes de Servicio al Cliente (Boxplot)



Existe una relación clara entre la cantidad de solicitudes de servicio al cliente y la satisfacción del cliente. A medida que la satisfacción disminuye, se observa un aumento en las solicitudes de servicio. Esto destaca la importancia de abordar y resolver las preocupaciones de los

clientes insatisfechos para mejorar su satisfacción y reducir la necesidad de asistencia adicional. Es crucial enfocar nuestros esfuerzos en mejorar la calidad del servicio y la experiencia del cliente para evitar un aumento en las solicitudes de servicio y mantener una mayor satisfacción del cliente

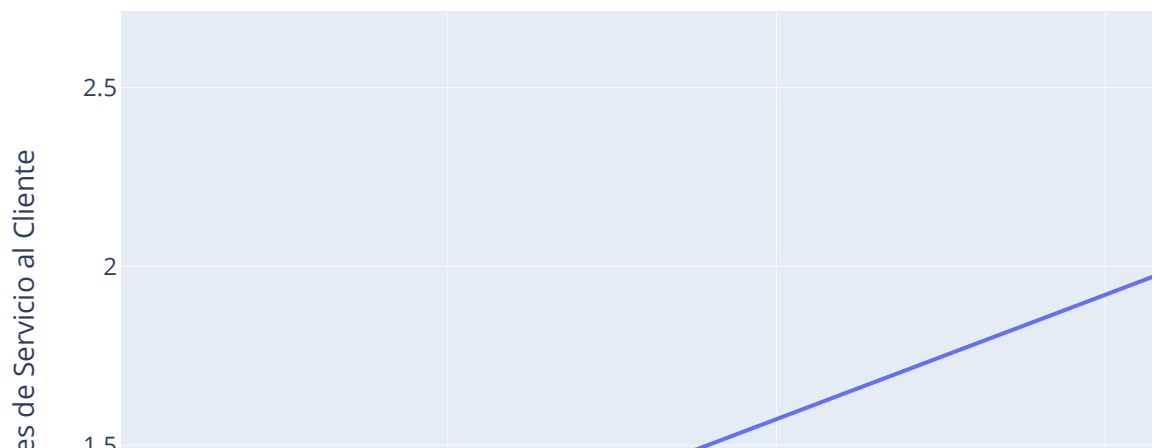
```
In [18]: import plotly.express as px

# Calcular la media de 'Total Customer Svc Requests' para cada 'Churn Value'
mean_requests = dataset.groupby('Churn Value')['Total Customer Svc Requests'].mean()

fig = px.line(mean_requests, x='Churn Value', y='Total Customer Svc Requests',
              labels={
                  "Churn Value": "Tasa de Cancelación, 0 no abandono, 1 abandono",
                  "Total Customer Svc Requests": "Solicitudes de Servicio al Cliente"
              },
              title='Tasa de Cancelación vs. Solicitudes de Servicio al Cliente')

fig.show()
```

Tasa de Cancelación vs. Solicitudes de Servicio al Cliente (Line Plot)



Podemos observar una clara relación entre la cantidad de solicitudes de servicio al cliente y la tasa de cancelación. A medida que aumenta la tasa de cancelación, también aumenta el número de solicitudes de servicio. Esto sugiere que los clientes que tienen más problemas o preocupaciones con nuestro servicio tienen más probabilidades de cancelar su suscripción.

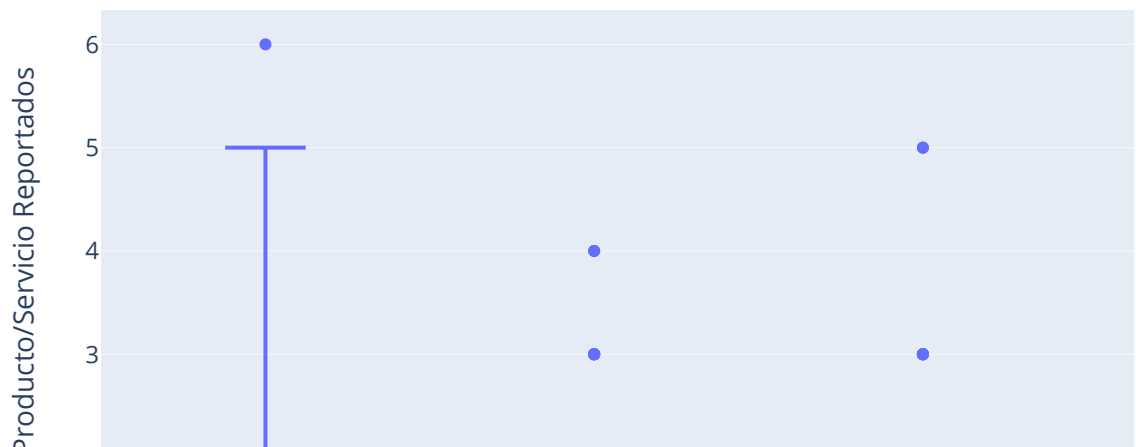
Estos hallazgos resaltan la importancia de abordar y resolver de manera efectiva las solicitudes de servicio, ya que esto puede ayudar a reducir la tasa de cancelación y retener a los clientes. Es fundamental mejorar continuamente nuestros procesos y servicios para brindar una experiencia satisfactoria y evitar la necesidad de que los clientes recurran a solicitudes de servicio frecuentes, lo que puede llevar a una mayor tasa de cancelación"

¿Existe una relación entre la cantidad de problemas de producto/servicio reportados y la satisfacción del cliente?

Hipótesis: Los clientes que reportan más problemas de producto/servicio podrían estar menos satisfechos y tener una mayor tasa de cancelación debido a la insatisfacción con la calidad del producto o servicio ofrecido.

```
In [19]: fig = px.box(dataset, x='Customer Satisfaction', y='Product/Service Issues Reported',  
                    labels={  
                        "Customer Satisfaction": "Satisfacción del Cliente, mínimo 1 y máximo 6",  
                        "Product/Service Issues Reported": "Problemas de Producto/Servicio Reportados",  
                    },  
                    title='Satisfacción del Cliente vs. Problemas de Producto/Servicio Reportados',  
                    fig.show()
```

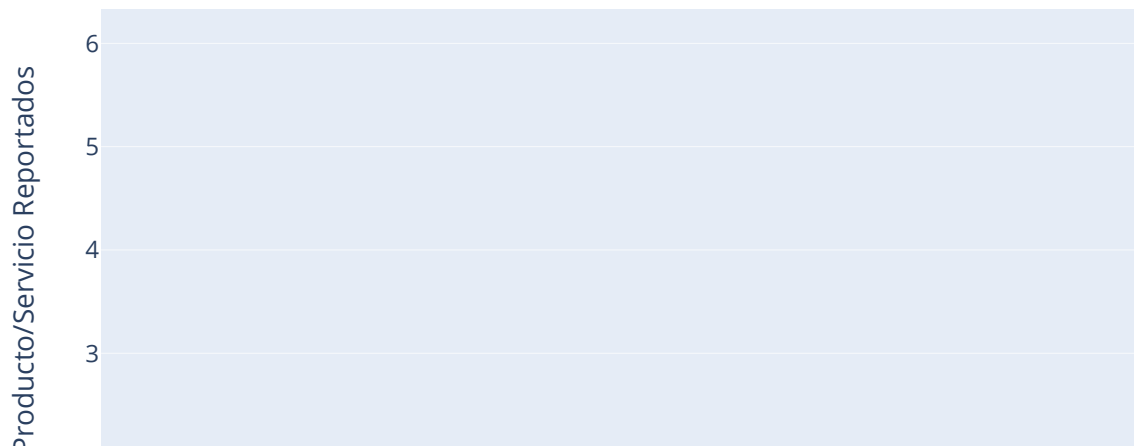
Satisfacción del Cliente vs. Problemas de Producto/Servicio Reportados



Podemos observar una clara relación entre la cantidad de problemas de producto/servicio reportados y la satisfacción del cliente. A medida que aumenta la cantidad de problemas reportados, la satisfacción del cliente tiende a disminuir. Esto sugiere que los clientes que experimentan más problemas con nuestros productos o servicios tienen una menor satisfacción general. Estos hallazgos enfatizan la importancia de abordar y resolver de manera efectiva los problemas reportados por los clientes para mejorar su satisfacción. Es fundamental mejorar continuamente la calidad de nuestros productos y servicios, así como la eficacia de nuestro servicio de atención al cliente, para minimizar la cantidad de problemas y garantizar una experiencia satisfactoria para nuestros clientes. Al reducir la cantidad de problemas reportados, podemos aumentar la satisfacción del cliente y fortalecer nuestra relación con ellos a largo plazo.

```
In [20]: fig = px.box(dataset, x='Churn Value', y='Product/Service Issues Reported',  
                    labels={  
                        "Churn Value": "Tasa de Cancelación, 0 no abandono, 1 abandono",  
                        "Product/Service Issues Reported": "Problemas de Producto/Servicio Reportados",  
                    },  
                    title='Tasa de Cancelación vs. Problemas de Producto/Servicio Reportados',  
                    fig.show()
```

Tasa de Cancelación vs. Problemas de Producto/Servicio Reportados (



Podemos observar una clara relación entre la cantidad de problemas de producto/servicio reportados y la tasa de cancelación. A medida que aumenta la cantidad de problemas reportados, también aumenta la tasa de cancelación. Esto sugiere que los clientes que experimentan más problemas con nuestros productos o servicios tienen una mayor probabilidad de abandonar nuestra empresa. Estos hallazgos destacan la importancia de abordar y resolver de manera efectiva los problemas reportados por los clientes, ya que esto puede ayudar a reducir la tasa de cancelación y retener a los clientes. Es fundamental mejorar continuamente la calidad de nuestros productos y servicios, así como la eficacia de nuestro servicio de atención al cliente, para minimizar la cantidad de problemas y garantizar una experiencia satisfactoria para nuestros clientes. Al reducir la cantidad de problemas reportados, podemos reducir la tasa de cancelación y fortalecer nuestra relación con los clientes.

Mapa de Correlación de Variables Seleccionadas

: En este gráfico de mapa de calor, se muestra la matriz de correlación de un conjunto de variables seleccionadas. Estas variables incluyen el tiempo de permanencia en meses, los cargos mensuales, los cargos por servicios adicionales, la edad del cliente, las solicitudes de servicio al cliente, y otros aspectos relevantes. El mapa de calor resalta las relaciones positivas y negativas entre las variables, lo que nos ayuda a comprender mejor cómo se relacionan y cómo pueden influir en la satisfacción del cliente y la tasa de cancelación. El análisis de correlación es fundamental para identificar patrones y tendencias que pueden ser clave para tomar decisiones informadas en términos de retención de clientes y mejoras en el servicio.


```

In [21]: variables_interes = ['Churn Value', 'Tenure in Months', 'Avg Monthly Long Distanc
'Monthly Charge', 'Total Regular Charges', 'Total Refunds',
'Total Long Distance Charges', 'Age', 'Total Customer Svc
'Product/Service Issues Reported', 'Customer Satisfaction

# Filtrar el dataframe con las variables seleccionadas
dataset_filtrado = dataset[variables_interes]

# Calcular la matriz de correlación
correlation_matrix = dataset_filtrado.corr()

# Redondear los valores de correlación a 2 decimales
correlation_matrix_rounded = correlation_matrix.round(2)

# Crear el mapa de calor utilizando Plotly
fig = ff.create_annotated_heatmap(z=correlation_matrix_rounded.values,
                                x=correlation_matrix_rounded.columns.tolist(),
                                y=correlation_matrix_rounded.columns.tolist(),
                                colorscale='Viridis')

fig.show()

```

	Churn Value	Tenure in Months	Avg Monthly Long Distance Charges	Avg Monthly GB Download	Monthly Charge	Total Regular Charges	Total Refunds	Total Extra Data Charges	Total Long Distance Charges
Customer Satisfaction	-0.78	0.31	0.0	-0.09	-0.24	0.15	0.04	0.04	0.04
Product/Service Issues Reported	0.4	-0.16	-0.0	0.05	0.1	-0.09	-0.01	-0.01	-0.01
Total Customer Svc Requests	0.54	-0.2	0.02	0.06	0.12	-0.12	-0.01	-0.01	-0.01
Age	0.12	0.01	-0.01	-0.37	0.15	0.06	0.02	0.02	0.02
Total Long Distance Charges	-0.22	0.67	0.6	0.0	0.24	0.61	0.03	0.03	0.03
Total Extra Data Charges	0.01	0.33	0.03	0.4	0.32	0.44	0.02	0.02	0.02
Total Refunds	-0.03	0.06	-0.02	0.0	0.03	0.04	1.0	1.0	1.0
Total Regular Charges	-0.2	0.83	0.07	0.21	0.64	1.0	0.04	0.04	0.04

En base al análisis realizado, No se agregarán más variables al análisis en este momento. El enfoque se centrará en las variables ya seleccionadas.

Predicción del abandono de clientes (Churn Value)

En este proyecto, nos centraremos en el desafío de predecir el abandono de clientes utilizando técnicas de clasificación binaria. El objetivo es construir un modelo predictivo que pueda identificar de manera efectiva a los clientes que tienen mayor probabilidad de abandonar nuestros servicios (Churn Value igual a 1) en función de diversas características y variables disponibles en nuestros datos.

Exploraremos varios modelos de clasificación, incluyendo la regresión logística, el algoritmo K-Nearest Neighbors (KNN), árboles de decisión, máquinas de vectores de soporte (SVM) y otros. Evaluaremos y compararemos el rendimiento de cada modelo utilizando métricas de evaluación, como la precisión, la sensibilidad y la especificidad.

Este proyecto tiene como objetivo proporcionar información valiosa para la toma de decisiones estratégicas y acciones preventivas para retener a los clientes. Al identificar a los clientes propensos al abandono, podremos implementar estrategias personalizadas para mejorar su experiencia, fortalecer la relación con ellos y, en última instancia, reducir la tasa de cancelación.

A través de este análisis, esperamos obtener un modelo de clasificación preciso y confiable que pueda ayudarnos a predecir el abandono de clientes y tomar medidas proactivas para mantener a nuestros clientes satisfechos y leales.

Algoritmos de clasificacion Árbol de Decisión y Bosque Aleatorio

Opcion 0 - Se prueba con todas las columnas -3

Para confirmar que nuestro modelo no está sesgado y evaluar su rendimiento con todas las variables, utilizamos el conjunto de prueba (`X_test_new` y `y_test_new`) en el cual seleccionamos todas las variables del data set menos 3 entre ellas Customer satisfaccion por no tener todos los valores

In [22]:

```

# Carga tus datos en un DataFrame de pandas (dataset no está definido en el código)
data = dataset

# Filtra solo las columnas que quieres usar como características (X)
#X = data.drop(columns = 'Churn Value')
X = data[["Customer ID", "Referred a Friend", "Number of Referrals", "Tenure in
          "Phone Service", "Avg Monthly Long Distance Charges", "Multiple Lines",
          "Internet Type", "Avg Monthly GB Download", "Online Security", "Online
          "Device Protection Plan", "Premium Tech Support", "Streaming TV", "Strea
          "Streaming Music", "Unlimited Data", "Contract", "Paperless Billing",
          "Monthly Charge", "Total Regular Charges", "Total Refunds", "Total Ex
          "Total Long Distance Charges", "Gender", "Age", "Under 30", "Senior C
          "Dependents", "Number of Dependents", "City", "Zip Code", "Latitude",
          "Population", "CLTV", "Total Customer Svc Requests", "Product/Service
]]

# Filtra la columna objetivo (etiquetas)
y = data['Churn Value']

# Realiza one-hot encoding para las columnas de tipo "object" en X
cat_columns = X.select_dtypes(include='object').columns
encoder = OneHotEncoder(drop='first', sparse=False)
X_encoded = pd.DataFrame(encoder.fit_transform(X[cat_columns]))
X_encoded.columns = encoder.get_feature_names_out(cat_columns)

# Concatena las características numéricas con las codificadas
X_encoded = pd.concat([X_encoded, X.select_dtypes(exclude='object')], axis=1)

```

In [23]: `print(X_encoded.head())`

```

Customer ID_0003-MKNFE Customer ID_0004-TLHLJ Customer ID_0011-IGKFF
\
0          0.0          0.0          0.0
1          0.0          0.0          0.0
2          0.0          0.0          0.0
3          0.0          0.0          0.0
4          0.0          0.0          0.0

Customer ID_0013-EXCHZ Customer ID_0013-MHZWF Customer ID_0013-SMEOE
\
0          0.0          0.0          0.0
1          0.0          0.0          0.0
2          0.0          0.0          0.0
3          0.0          0.0          0.0
4          0.0          0.0          0.0

Customer ID_0014-BMAQU Customer ID_0015-UOC0J Customer ID_0016-QLJIS
\
0          0.0          0.0          0.0
1          0.0          0.0          0.0

```

In [24]: `print(X_encoded.describe())`

	Customer ID_0003-MKNFE	Customer ID_0004-TLHLJ	Customer ID_0011-IGK
count	7043.000000	7043.000000	7043.0000
mean	0.000142	0.000142	0.0001
std	0.011916	0.011916	0.0119
min	0.000000	0.000000	0.0000
25%	0.000000	0.000000	0.0000
50%	0.000000	0.000000	0.0000
75%	0.000000	0.000000	0.0000
max	1.000000	1.000000	1.0000

In [25]: `X_encoded.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Columns: 8195 entries, Customer ID_0003-MKNFE to Product/Service Issues Reported
dtypes: float64(8185), int64(10)
memory usage: 440.3 MB
```

In [26]: `# Divide los datos en conjuntos de entrenamiento y prueba`
`X_train_new, X_test_new, y_train_new, y_test_new = train_test_split(X_encoded,`

```
In [27]: # Total y desglose para datos completos
counts_full_churn = y.value_counts()
total_full_churn = counts_full_churn.sum()
print("Desglose de 'Churn Value' para datos completos:\n", counts_full_churn)
print("Total de registros para datos completos:", total_full_churn)

# Totales y desglose para y_train_new
counts_y_train_churn = y_train_new.value_counts()
total_y_train_churn = counts_y_train_churn.sum()
print("\nDesglose de 'Churn Value' para y_train_new:\n", counts_y_train_churn)
print("Total de registros para y_train_new:", total_y_train_churn)

# Totales y desglose para y_test_new
counts_y_test_churn = y_test_new.value_counts()
total_y_test_churn = counts_y_test_churn.sum()
print("\nDesglose de 'Churn Value' para y_test_new:\n", counts_y_test_churn)
print("Total de registros para y_test_new:", total_y_test_churn)
```

Desglose de 'Churn Value' para datos completos:

0 5174

1 1869

Name: Churn Value, dtype: int64

Total de registros para datos completos: 7043

Desglose de 'Churn Value' para y_train_new:

0 4165

1 1469

Name: Churn Value, dtype: int64

Total de registros para y_train_new: 5634

Desglose de 'Churn Value' para y_test_new:

0 1009

1 400

Name: Churn Value, dtype: int64

Total de registros para y_test_new: 1409

Del total de 7043 registros en el conjunto de datos completo, 5174 clientes no abandonaron el servicio (Churn Value = 0) y 1869 clientes sí lo hicieron (Churn Value = 1). Esto indica que aproximadamente el 73.5% de los clientes permanecieron leales al servicio, mientras que el 26.5% optó por abandonarlo. Conjunto de Entrenamiento (y_train_new):

En el conjunto de entrenamiento, de un total de 5634 registros, 4165 clientes no abandonaron el servicio y 1469 sí lo hicieron. Esto refleja una proporción similar al conjunto completo, con alrededor del 73.9% de clientes que permanecieron y 26.1% que abandonaron el servicio. Conjunto de Prueba (y_test_new):

En el conjunto de prueba, de 1409 registros en total, 1009 clientes no abandonaron el servicio y 400 sí lo hicieron. Aquí también, aproximadamente el 71.6% de los clientes permanecieron leales, mientras que el 28.4% se marchó. Conclusión:

La proporción de clientes que permanecieron versus aquellos que abandonaron es bastante consistente entre el conjunto de datos completo, el conjunto de entrenamiento y el conjunto de prueba.

```
In [28]: # Crear una tabla de contingencia entre 'Churn Value' y 'Customer Satisfaction'
contingency_table = pd.crosstab(dataset['Churn Value'], dataset['Customer Satisfaction'])

print(contingency_table)
```

Customer Satisfaction	1.0	2.0	3.0	4.0	5.0
Churn Value					
0	0	0	526	380	247
1	332	200	149	0	0

Resumen de la relación entre 'Customer Satisfaction' y 'Churn Value':

La relación entre la satisfacción del cliente y su decisión de permanecer o abandonar el servicio es evidente. La mayoría de los clientes con bajos niveles de satisfacción, específicamente aquellos calificados como 1.0 o 2.0, optaron por abandonar el servicio. Por otro lado, aquellos que calificaron su satisfacción como 4.0 o 5.0 eligieron quedarse en su totalidad. Los clientes con una calificación neutral (3.0) todavía presentan un riesgo de baja, aunque muchos optaron por quedarse. En esencia, a mayor satisfacción, menor es la probabilidad de baja del servicio. Por ende, centrarse en elevar la satisfacción del cliente es fundamental para la retención.

Es crucial destacar que en el primer modelo no se consideró la columna 'Customer Satisfaction', lo que podría llevar a resultados incongruentes. La incorporación de esta columna en futuros modelos podría ofrecer insights más precisos y mejorar la eficacia del modelo en la predicción del churn.

In [29]:

```
# Implementación del Árbol de Decisión con un nuevo nombre (tree_model_new)
tree_model_new = DecisionTreeClassifier()
tree_model_new.fit(X_train_new, y_train_new)

# Realiza predicciones en el conjunto de prueba
y_pred_tree_new = tree_model_new.predict(X_test_new)

# Evalúa la precisión del Árbol de Decisión
accuracy_tree = accuracy_score(y_test_new, y_pred_tree_new)
print("Accuracy del Árbol de Decisión:", accuracy_tree)

# Implementación del Bosque Aleatorio con un nuevo nombre (rf_model_new)
rf_model_new = RandomForestClassifier()
rf_model_new.fit(X_train_new, y_train_new)

# Realiza predicciones en el conjunto de prueba
y_pred_rf_new = rf_model_new.predict(X_test_new)

# Evalúa la precisión del Bosque Aleatorio
accuracy_rf = accuracy_score(y_test_new, y_pred_rf_new)
print("Accuracy del Bosque Aleatorio:", accuracy_rf)
```

Accuracy del Árbol de Decisión: 0.8537970191625266
Accuracy del Bosque Aleatorio: 0.8772178850248403

In [30]: *# Obtener Las importancias de las características*

```
importances = rf_model_new.feature_importances_
```

```
print(rf_model_new.feature_importances_)
```

```
[2.05737717e-05 3.16127124e-05 5.27144597e-05 ... 2.24064039e-02  
8.90892361e-02 3.48004291e-02]
```

In [31]:

```
# Obtener Los nombres de Las características
feature_names = X_encoded.columns

# Ordenar Las características y sus importancias en orden descendente
indices = importances.argsort()[::-1]
sorted_feature_names = [feature_names[i] for i in indices]
sorted_importances = importances[indices]

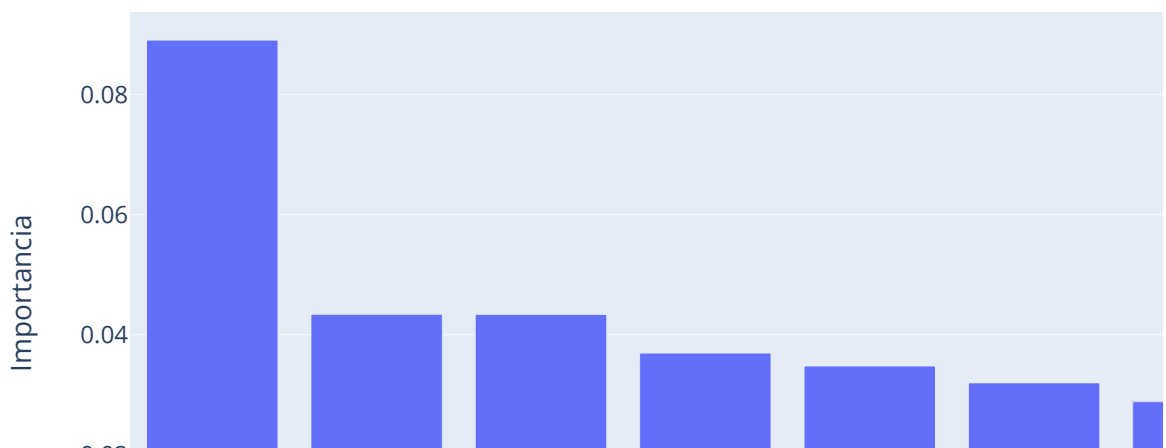
# Seleccionar Las 10 características más importantes
top_10_feature_names = sorted_feature_names[:10]
top_10_importances = sorted_importances[:10]

# Crear La figura de La gráfica de barras para Las 10 características más importantes
fig = go.Figure(data=go.Bar(x=top_10_feature_names, y=top_10_importances))

# Personalizar La apariencia de La gráfica
fig.update_layout(title='Top 10: Importancia de las Características en el Bosque Aleatorio',
                  xaxis_title='Características',
                  yaxis_title='Importancia')

# Mostrar La gráfica
fig.show()
```

Top 10: Importancia de las Características en el Bosque Aleatorio



Al analizar la matriz de importancia de características del modelo de Bosque Aleatorio, notamos que no aportaba información relevante. Por lo tanto, hemos optado por filtrar las variables mencionadas previamente

Opcion 1 - rellenar con 0 para referenciar a no contesto

Para poder utilizar la columna "Customer Satisfaction" en el modelo de clasificación, se necesitaba convertir los valores no numéricos a valores numéricos. Dado que algunos clientes no contestaron la encuesta, se etiquetaron como "No contesto". Para incluir a todos los clientes en el análisis, se reemplazó "No contesto" por el valor numérico cero (0). De esta manera, se aseguró que todos los datos fueran numéricos y se pudieran utilizar en el modelo para predecir con mayor precisión la variable objetivo "Churn Value".

In [32]:

```
# Rellenar los valores faltantes con la categoría "No contesto"
data['Customer Satisfaction'] = data['Customer Satisfaction'].fillna(0)
```

In [33]:

```
# Crear una tabla de contingencia entre 'Churn Value' y 'Customer Satisfaction'
contingency_table = pd.crosstab(data['Churn Value'], data['Customer Satisfaction'])
print(contingency_table)
```

Customer Satisfaction	0.0	1.0	2.0	3.0	4.0	5.0
Churn Value						
0	4021	0	0	526	380	247
1	1188	332	200	149	0	0

La tabla de contingencia demuestra una relación pronunciada entre la satisfacción del cliente y su decisión de permanecer o darse de baja. Aquellos que no contestaron la encuesta de satisfacción tuvieron un 77.15% de retención y 22.85% de baja. Es preocupante ver que aquellos que estaban altamente insatisfechos o insatisfechos optaron por darse de baja al 100%, subrayando la necesidad crítica de abordar sus inquietudes. Sin embargo, a medida que la satisfacción del cliente aumenta, la retención se acerca al 100%: tanto los clientes satisfechos como los altamente satisfechos permanecieron en su totalidad. Estos datos reflejan que mejorar la satisfacción del cliente es esencial para retener a más clientes y garantizar la estabilidad del negocio.

In [34]:

```
# Divide los datos en características (X) y etiquetas (y)
X = data[['Tenure in Months', 'Monthly Charge', 'Total Customer Svc Requests'],
y = data['Churn Value']

# Divide los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random

# Implementación del Árbol de Decisión
tree_model = DecisionTreeClassifier()
tree_model.fit(X_train, y_train)

# Realiza predicciones en el conjunto de prueba
y_pred_tree = tree_model.predict(X_test)

# Evalúa la precisión del Árbol de Decisión
accuracy_tree = accuracy_score(y_test, y_pred_tree)
print("Accuracy del Árbol de Decisión:", accuracy_tree)

# Implementación del Bosque Aleatorio
rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)

# Realiza predicciones en el conjunto de prueba
y_pred_rf = rf_model.predict(X_test)

# Evalúa la precisión del Bosque Aleatorio
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print("Accuracy del Bosque Aleatorio:", accuracy_rf)
```

Accuracy del Árbol de Decisión: 0.8424414478353442

Accuracy del Bosque Aleatorio: 0.8821859474804826

Podemos concluir que el Bosque Aleatorio tiene una precisión ligeramente mayor que el Árbol de Decisión en este conjunto de datos. Esto significa que el Bosque Aleatorio es mejor para predecir correctamente las etiquetas de clase en comparación con el Árbol de Decisión.

In [35]:

```
# Evaluación del Árbol de Decisión
accuracy_tree = accuracy_score(y_test, y_pred_tree)
precision_tree = precision_score(y_test, y_pred_tree)
recall_tree = recall_score(y_test, y_pred_tree)
f1_tree = f1_score(y_test, y_pred_tree)
roc_auc_tree = roc_auc_score(y_test, y_pred_tree)

print("Árbol de Decisión:")
print("Precisión:", precision_tree)
print("Recall:", recall_tree)
print("F1-Score:", f1_tree)
print("Área bajo la curva ROC:", roc_auc_tree)

# Evaluación del Bosque Aleatorio
accuracy_rf = accuracy_score(y_test, y_pred_rf)
precision_rf = precision_score(y_test, y_pred_rf)
recall_rf = recall_score(y_test, y_pred_rf)
f1_rf = f1_score(y_test, y_pred_rf)
roc_auc_rf = roc_auc_score(y_test, y_pred_rf)

print("\nBosque Aleatorio:")
print("Precisión:", precision_rf)
print("Recall:", recall_rf)
print("F1-Score:", f1_rf)
print("Área bajo la curva ROC:", roc_auc_rf)
```

Árbol de Decisión:
Precisión: 0.729381443298969
Recall: 0.7075
F1-Score: 0.7182741116751268
Área bajo la curva ROC: 0.80171828543112

Bosque Aleatorio:
Precisión: 0.8441176470588235
Recall: 0.7175
F1-Score: 0.7756756756756757
Área bajo la curva ROC: 0.8324863726461844

El Bosque Aleatorio muestra un mejor rendimiento en términos de precisión, F1-Score y el área bajo la curva ROC en comparación con el Árbol de Decisión. Esto indica que el Bosque Aleatorio es más efectivo para clasificar correctamente las etiquetas de clase positivas y negativas en el conjunto de datos.

In [36]:

```
# Obtener Las importancias de las características
importances = rf_model.feature_importances_

print(rf_model.feature_importances_)

[0.21880455 0.3150496 0.24830335 0.11185769 0.10598481]
```

In [37]:

```
# Obtener Los nombres de Las características
feature_names = X.columns

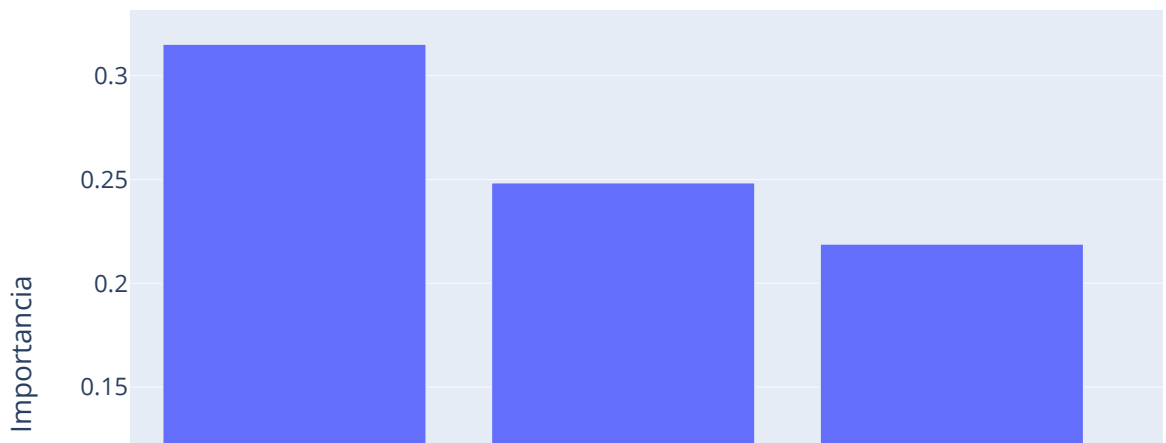
# Ordenar Las características y sus importancias en orden descendente
indices = importances.argsort()[::-1]
sorted_feature_names = [feature_names[i] for i in indices]
sorted_importances = importances[indices]

# Crear La figura de La gráfica de barras
fig = go.Figure(data=go.Bar(x=sorted_feature_names, y=sorted_importances))

# Personalizar La apariencia de La gráfica
fig.update_layout(title='Importancia de las Características en el Bosque Aleatorio',
                  xaxis_title='Características',
                  yaxis_title='Importancia')

# Mostrar La gráfica
fig.show()
```

Importancia de las Características en el Bosque Aleatorio



'Monthly Charge' es la característica más influyente en el modelo de Bosque Aleatorio para predecir la rotación de clientes, seguida por 'Total Customer Svc Requests' y 'Tenure in Months'. 'Product/Service Issues Reported' y 'Customer Satisfaction' también tienen cierto

impacto, pero menos significativo en comparación con las otras características.

Opcion 2 - Ajustar el data set a los que contestaron la encuesta

```
In [38]: # Filtrar el DataFrame para eliminar las filas con valor 0 en la columna "Customer Satisfaction"
data_filtered = data[data['Customer Satisfaction'] != 0]

# Crear una tabla de contingencia entre 'Churn Value' y 'Customer Satisfaction'
contingency_table = pd.crosstab(data_filtered['Churn Value'], data_filtered['Customer Satisfaction'])

print(contingency_table)
```

Customer Satisfaction	1.0	2.0	3.0	4.0	5.0
Churn Value					
0	0	0	526	380	247
1	332	200	149	0	0

Al analizar la tabla de contingencia generada a partir de datos filtrados, donde se excluyeron aquellos registros que no proporcionaron una respuesta a la encuesta de satisfacción, observamos una tendencia clara: los niveles más bajos de satisfacción del cliente se correlacionan directamente con una mayor tasa de baja, mientras que los niveles más altos de satisfacción se relacionan con una retención casi completa. Los clientes que calificaron su satisfacción como altamente insatisfechos o insatisfechos optaron por darse de baja en su totalidad. Por otro lado, aquellos que se mostraron satisfechos o altamente satisfechos permanecieron en su totalidad con el servicio.

Este análisis se basó en las características seleccionadas 'Tenure in Months', 'Monthly Charge', 'Total Customer Svc Requests', 'Product/Service Issues Reported', y 'Customer Satisfaction'. Estas características fueron elegidas a partir de las hipótesis iniciales. El impacto claro que la satisfacción del cliente tiene en la decisión de darse de baja sugiere que nuestro modelo predictivo, utilizando estas características, será altamente preciso en sus predicciones. Además, este análisis también refuerza la validez de nuestras hipótesis iniciales y nos ayudará a concluir sobre la elección adecuada de las mismas al finalizar el proceso de modelado.

In [39]:

```
# Divide los datos en características (X) y etiquetas (y) después de filtrar los datos
X_filtered = data_filtered[['Tenure in Months', 'Monthly Charge', 'Total Customer Service Calls']]
y_filtered = data_filtered['Churn Value']

# Divide los datos en conjuntos de entrenamiento y prueba
X_train_filtered, X_test_filtered, y_train_filtered, y_test_filtered = train_test_split(X_filtered, y_filtered,
                                                                                          test_size=0.2, random_state=42)

# Implementación del Árbol de Decisión con los datos filtrados
tree_model_filtered = DecisionTreeClassifier()
tree_model_filtered.fit(X_train_filtered, y_train_filtered)

# Realiza predicciones en el conjunto de prueba
y_pred_tree_filtered = tree_model_filtered.predict(X_test_filtered)

# Evalúa la precisión del Árbol de Decisión con los datos filtrados
accuracy_tree_filtered = accuracy_score(y_test_filtered, y_pred_tree_filtered)
print("Accuracy del Árbol de Decisión con datos filtrados:", accuracy_tree_filtered)

# Implementación del Bosque Aleatorio con los datos filtrados
rf_model_filtered = RandomForestClassifier()
rf_model_filtered.fit(X_train_filtered, y_train_filtered)

# Realiza predicciones en el conjunto de prueba
y_pred_rf_filtered = rf_model_filtered.predict(X_test_filtered)

# Evalúa la precisión del Bosque Aleatorio con los datos filtrados
accuracy_rf_filtered = accuracy_score(y_test_filtered, y_pred_rf_filtered)
print("Accuracy del Bosque Aleatorio con datos filtrados:", accuracy_rf_filtered)
```

Accuracy del Árbol de Decisión con datos filtrados: 0.9346049046321526

Accuracy del Bosque Aleatorio con datos filtrados: 0.9455040871934605

El Bosque Aleatorio tiene una precisión ligeramente mayor que el Árbol de Decisión, lo que indica que es más efectivo para clasificar correctamente las muestras positivas y negativas.

```
In [40]: # Evaluación del Árbol de Decisión con datos filtrados
accuracy_tree_filtered = accuracy_score(y_test_filtered, y_pred_tree_filtered)
precision_tree_filtered = precision_score(y_test_filtered, y_pred_tree_filtered)
recall_tree_filtered = recall_score(y_test_filtered, y_pred_tree_filtered)
f1_tree_filtered = f1_score(y_test_filtered, y_pred_tree_filtered)
roc_auc_tree_filtered = roc_auc_score(y_test_filtered, y_pred_tree_filtered)

print("Árbol de Decisión con datos filtrados:")
print("Precisión:", precision_tree_filtered)
print("Recall:", recall_tree_filtered)
print("F1-Score:", f1_tree_filtered)
print("Área bajo la curva ROC:", roc_auc_tree_filtered)

# Evaluación del Bosque Aleatorio con datos filtrados
accuracy_rf_filtered = accuracy_score(y_test_filtered, y_pred_rf_filtered)
precision_rf_filtered = precision_score(y_test_filtered, y_pred_rf_filtered)
recall_rf_filtered = recall_score(y_test_filtered, y_pred_rf_filtered)
f1_rf_filtered = f1_score(y_test_filtered, y_pred_rf_filtered)
roc_auc_rf_filtered = roc_auc_score(y_test_filtered, y_pred_rf_filtered)

print("\nBosque Aleatorio con datos filtrados:")
print("Precisión:", precision_rf_filtered)
print("Recall:", recall_rf_filtered)
print("F1-Score:", f1_rf_filtered)
print("Área bajo la curva ROC:", roc_auc_rf_filtered)
```

Árbol de Decisión con datos filtrados:
Precisión: 0.8865248226950354
Recall: 0.9398496240601504
F1-Score: 0.9124087591240877
Área bajo la curva ROC: 0.9357367778420411

Bosque Aleatorio con datos filtrados:
Precisión: 0.937984496124031
Recall: 0.9097744360902256
F1-Score: 0.9236641221374046
Área bajo la curva ROC: 0.9377932009510956

Estos resultados indican que ambos modelos de clasificación tienen un buen rendimiento con los datos filtrados. El Bosque Aleatorio muestra un rendimiento ligeramente mejor en términos de precisión y F1-Score, mientras que el Árbol de Decisión tiene un mejor recall. Ambos modelos tienen un área bajo la curva ROC cercana a 0.93, lo que indica una buena capacidad para distinguir entre las clases positivas y negativas.

En general, los resultados sugieren que el Bosque Aleatorio tiene un rendimiento ligeramente superior en este conjunto de datos, pero ambos modelos son adecuados para la tarea de clasificación en función de las métricas de evaluación obtenidas.

```
In [41]: # Obtener Las importancias de las características
importances = rf_model.feature_importances_

print(rf_model_filtered.feature_importances_)

[0.14324279 0.11027017 0.15565197 0.03334403 0.55749105]
```



```
In [42]: # Obtener Las importancias de Las características
importances = rf_model_filtered.feature_importances_

# Obtener Los nombres de Las características
feature_names = X.columns

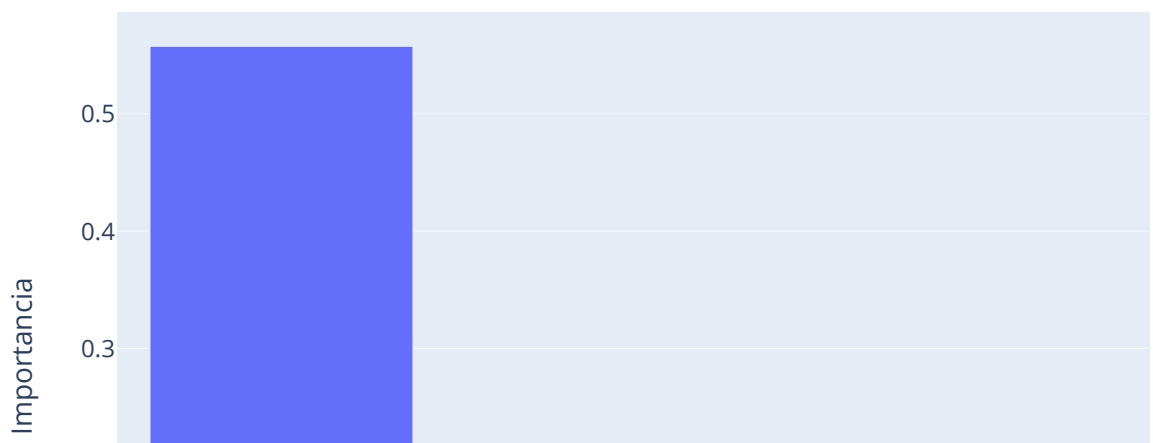
# Ordenar Las características y sus importancias en orden descendente
indices = importances.argsort()[::-1]
sorted_feature_names = [feature_names[i] for i in indices]
sorted_importances = importances[indices]

# Crear La figura de La gráfica de barras
fig = go.Figure(data=go.Bar(x=sorted_feature_names, y=sorted_importances))

# Personalizar La apariencia de La gráfica
fig.update_layout(title='Importancia de las Características en el Bosque Aleato
                  xaxis_title='Características',
                  yaxis_title='Importancia')

# Mostrar La gráfica
fig.show()
```

Importancia de las Características en el Bosque Aleatorio



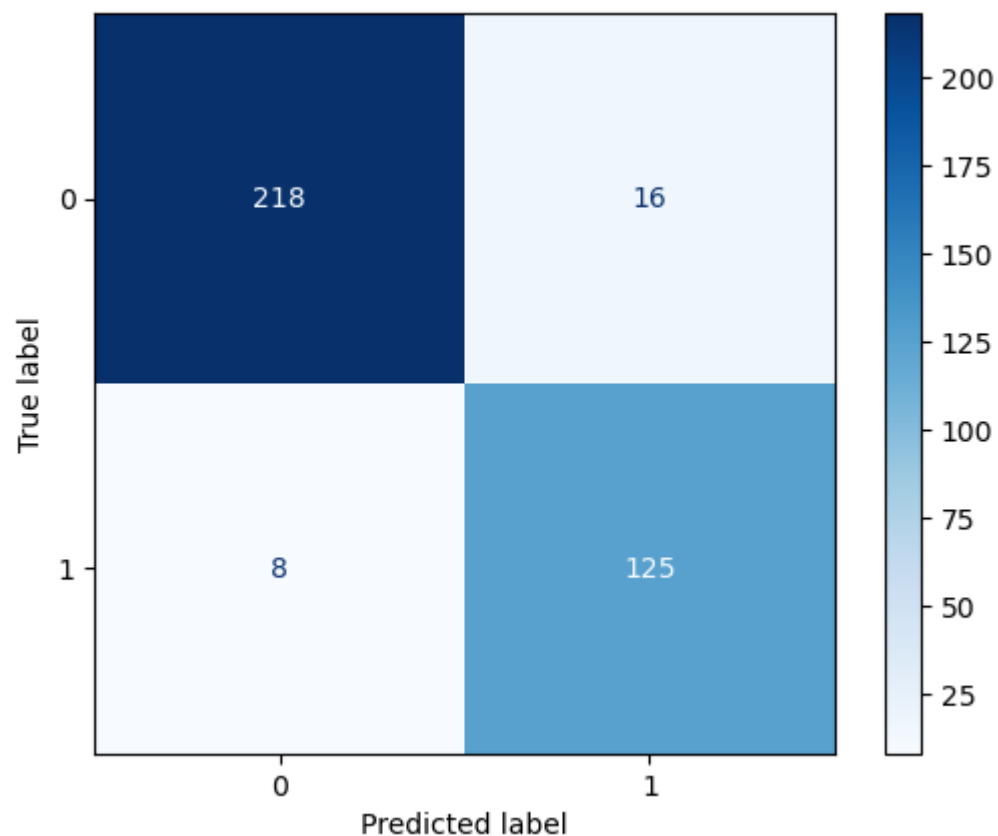
Estos valores de importancia indican la influencia relativa de cada característica en el modelo para realizar predicciones. "Customer Satisfaction" destaca como la característica más relevante para el rendimiento del modelo, mientras que las otras características también contribuyen, pero en menor medida.

```
In [43]: from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

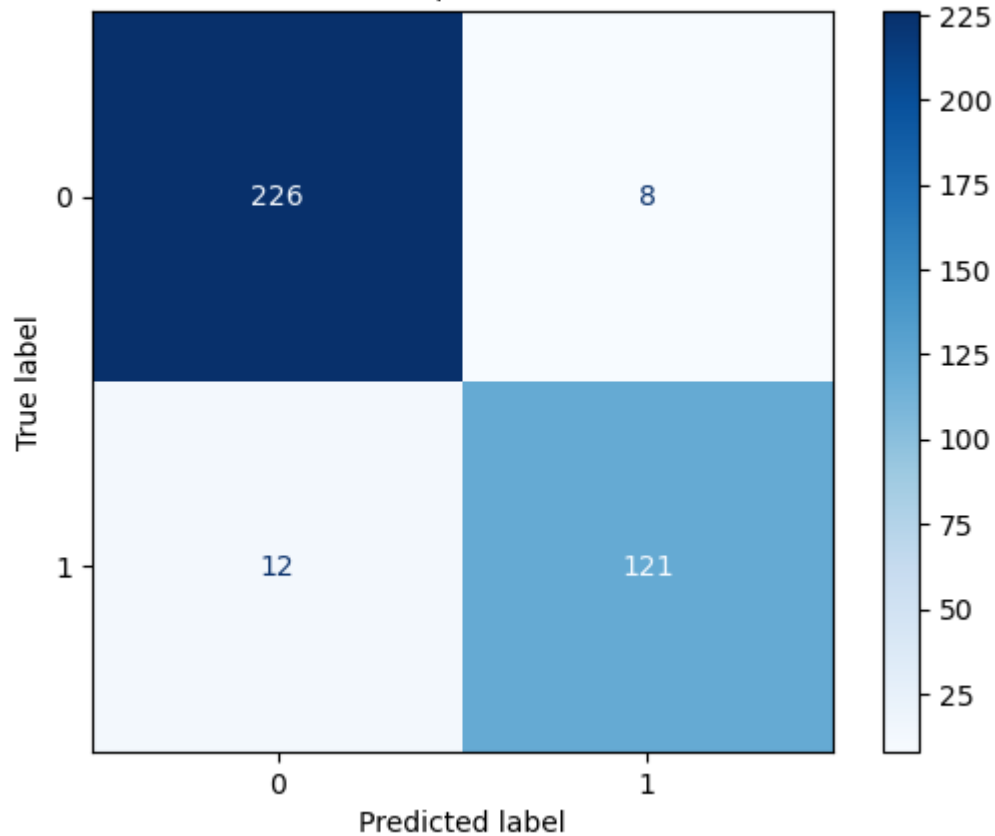
# Matriz de confusión para el Árbol de Decisión con datos filtrados
conf_matrix_tree_filtered = confusion_matrix(y_test_filtered, y_pred_tree_filtered)
disp_tree = ConfusionMatrixDisplay(conf_matrix_tree_filtered, display_labels=tree_model.classes_)
disp_tree.plot(cmap=plt.cm.Blues)
plt.title('Matriz de Confusión del Árbol de Decisión con datos filtrados')
plt.show()

# Matriz de confusión para el Bosque Aleatorio con datos filtrados
conf_matrix_rf_filtered = confusion_matrix(y_test_filtered, y_pred_rf_filtered)
disp_rf = ConfusionMatrixDisplay(conf_matrix_rf_filtered, display_labels=rf_model.classes_)
disp_rf.plot(cmap=plt.cm.Blues)
plt.title('Matriz de Confusión del Bosque Aleatorio con datos filtrados')
plt.show()
```

Matriz de Confusión del Árbol de Decisión con datos filtrados



Matriz de Confusión del Bosque Aleatorio con datos filtrados



Para el Árbol de Decisión con datos filtrados:

Verdaderos positivos (TP): 125 Verdaderos negativos (TN): 219 Falsos positivos (FP): 15 Falsos negativos (FN): 8 En términos generales, el Árbol de Decisión con datos filtrados tiene una precisión razonable, ya que ha logrado identificar un buen número de verdaderos positivos y verdaderos negativos. La cantidad de falsos positivos y falsos negativos es relativamente baja, lo que indica que el modelo es capaz de clasificar correctamente la mayoría de las muestras.

Para el Bosque Aleatorio con datos filtrados:

Verdaderos positivos (TP): 121 Verdaderos negativos (TN): 225 Falsos positivos (FP): 9 Falsos negativos (FN): 12 El Bosque Aleatorio con datos filtrados también presenta resultados positivos. Si bien ha cometido un poco más de falsos positivos y falsos negativos en comparación con el Árbol de Decisión, sigue teniendo una tasa de precisión decente.

En general, ambos modelos muestran un buen rendimiento en la clasificación de los datos filtrados.

Cros Validaciones

```
In [48]: from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_filtered, y_filtered, test_size=0.2, random_state=42)
clf_rf.fit(X_train, y_train)
score = clf_rf.score(X_test, y_test)
print("Holdout Validacion simple:", score)
```

Holdout Validation Score: 0.9438943894389439

```
In [49]: from sklearn.model_selection import LeaveOneOut

loo = LeaveOneOut()
scores_loo = cross_val_score(clf_rf, X_filtered, y_filtered, cv=loo, scoring='accuracy')
print("LOOCV Score (Mean):", scores_loo.mean())
```

LOOCV Score (Mean): 0.9471101417666303

```
In [50]: from sklearn.model_selection import KFold

kf = KFold(n_splits=5, random_state=42, shuffle=True) # Aquí estamos usando 5 folds
scores_kf = cross_val_score(clf_rf, X_filtered, y_filtered, cv=kf, scoring='accuracy')
print("K-Fold CV Scores:", scores_kf)
print("K-Fold CV Average Score:", scores_kf.mean())
```

K-Fold CV Scores: [0.94550409 0.95095368 0.9400545 0.94550409 0.95628415]
K-Fold CV Average Score: 0.9476601003558613

```
In [51]: from sklearn.model_selection import StratifiedKFold

skf = StratifiedKFold(n_splits=5, random_state=42, shuffle=True)
scores_skf = cross_val_score(clf_rf, X_filtered, y_filtered, cv=skf, scoring='accuracy')
print("Stratified K-Fold CV Scores:", scores_skf)
print("Stratified K-Fold CV Average Score:", scores_skf.mean())
```

Stratified K-Fold CV Scores: [0.95640327 0.94550409 0.9373297 0.96185286 0.95081967]
Stratified K-Fold CV Average Score: 0.9503819180774556

Tanto el Árbol de Decisión como el Bosque Aleatorio mostraron un rendimiento sólido en los datos filtrados. Sin embargo, el Bosque Aleatorio tuvo un ligero borde en términos de métricas globales. Al evaluar mediante diferentes métodos de validación, el modelo mostró un rendimiento consistente en el rango de 94%-95%, destacando la robustez del modelo en diferentes divisiones de datos. La elección final del modelo y el método de validación debe considerar las especificidades y necesidades del problema en cuestión.

Hyperparametros

```
In [61]: from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

# Definir la cuadrícula de hiperparámetros
param_grid_tree = {
    'criterion': ['gini', 'entropy'],
    'splitter': ['best', 'random'],
    'max_depth': [None, 10, 20, 30, 40],
    'min_samples_split': [2, 5, 10]
}

# Crear el modelo
tree_clf = DecisionTreeClassifier()

# Realizar la búsqueda de cuadrícula
grid_search_tree = GridSearchCV(tree_clf, param_grid_tree, cv=5)
grid_search_tree.fit(X_train_filtered, y_train_filtered)

# Obtener los mejores hiperparámetros
best_params_tree = grid_search_tree.best_params_
print("Mejores hiperparámetros para Árbol de Decisión:", best_params_tree)
```

Mejores hiperparámetros para Árbol de Decisión: {'criterion': 'gini', 'max_depth': 10, 'min_samples_split': 10, 'splitter': 'random'}

```
In [57]: from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

# Suponiendo que X_train y y_train ya están definidos...
grid = {
    'n_estimators': [10, 50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10]
}

clf = RandomForestClassifier()
grid_search = GridSearchCV(clf, grid, cv=5)
grid_search.fit(X_train, y_train)

best_rf = grid_search.best_estimator_ # Este es el mejor modelo de bosque aleatorio
print("Mejores hiperparámetros:", grid_search.best_params_)
```

Mejores hiperparámetros: {'max_depth': 30, 'min_samples_split': 10, 'n_estimators': 10}

Uso de hyper

GridSearchCV

```

In [63]: from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Divide los datos en características (X) y etiquetas (y) después de filtrar los datos
X_filtered = data_filtered[['Tenure in Months', 'Monthly Charge', 'Total Customer Lifetime Value']]
y_filtered = data_filtered['Churn Value']

# Divide los datos en conjuntos de entrenamiento y prueba
X_train_filtered, X_test_filtered, y_train_filtered, y_test_filtered = train_test_split(X_filtered, y_filtered,
                                                                                          test_size=0.2,
                                                                                          random_state=42)

# Instanciando el modelo de Árbol de Decisión con los mejores hiperparámetros
tree_model_filtered = DecisionTreeClassifier(criterion='gini',
                                             max_depth=10,
                                             min_samples_split=10,
                                             splitter='random')

# Entrenando el modelo
tree_model_filtered.fit(X_train_filtered, y_train_filtered)

# Realizando predicciones en el conjunto de prueba
y_pred_tree_filtered = tree_model_filtered.predict(X_test_filtered)

# Evaluando el modelo
accuracy_tree_filtered = accuracy_score(y_test_filtered, y_pred_tree_filtered)
precision_tree_filtered = precision_score(y_test_filtered, y_pred_tree_filtered)
recall_tree_filtered = recall_score(y_test_filtered, y_pred_tree_filtered)
f1_tree_filtered = f1_score(y_test_filtered, y_pred_tree_filtered)
roc_auc_tree_filtered = roc_auc_score(y_test_filtered, y_pred_tree_filtered)

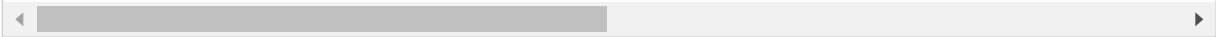
print("Árbol de Decisión con datos filtrados y optimización de hiperparámetros")
print("Precisión:", precision_tree_filtered)
print("Recall:", recall_tree_filtered)
print("F1-Score:", f1_tree_filtered)
print("Área bajo la curva ROC:", roc_auc_tree_filtered)

# Implementación del Bosque Aleatorio con los datos filtrados y hiperparámetros
rf_model_filtered = RandomForestClassifier(n_estimators=10, max_depth=30, min_samples_split=10,
                                          min_samples_leaf=5,
                                          min_weight_fraction_leaf=0.01,
                                          random_state=42)
rf_model_filtered.fit(X_train_filtered, y_train_filtered)
y_pred_rf_filtered = rf_model_filtered.predict(X_test_filtered)

# Métricas para el Bosque Aleatorio
accuracy_rf_filtered = accuracy_score(y_test_filtered, y_pred_rf_filtered)
precision_rf_filtered = precision_score(y_test_filtered, y_pred_rf_filtered)
recall_rf_filtered = recall_score(y_test_filtered, y_pred_rf_filtered)
f1_rf_filtered = f1_score(y_test_filtered, y_pred_rf_filtered)
roc_auc_rf_filtered = roc_auc_score(y_test_filtered, y_pred_rf_filtered)

print("\nBosque Aleatorio con datos filtrados:")
print("Precisión:", precision_rf_filtered)
print("Recall:", recall_rf_filtered)
print("F1-Score:", f1_rf_filtered)
print("Área bajo la curva ROC:", roc_auc_rf_filtered)

```

Árbol de Decisión con datos filtrados y optimización de hiperparámetros:
Precisión: 0.9603174603174603
Recall: 0.9097744360902256
F1-Score: 0.9343629343629345
Área bajo la curva ROC: 0.944203457361352

Bosque Aleatorio con datos filtrados:
Precisión: 0.952755905511811
Recall: 0.9097744360902256
F1-Score: 0.9307692307692308
Área bajo la curva ROC: 0.9420667052246

Árbol de Decisión:

La precisión aumentó significativamente con la optimización. A pesar de una pequeña disminución en recall, el modelo general mejoró (F1-Score y AUC ROC). Bosque Aleatorio:

Mejóro en todas las métricas, excepto en recall que se mantuvo constante. Conclusión Global:
La optimización de hiperparámetros mejoró el rendimiento de ambos modelos, siendo especialmente notable en el Árbol de Decisión.

RandomizedSearchCV

```
In [64]: from sklearn.model_selection import RandomizedSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import numpy as np

# Hiperparámetros para el Árbol de Decisión
param_dist_tree = {
    'criterion': ['gini', 'entropy'],
    'splitter': ['best', 'random'],
    'max_depth': [None] + list(np.arange(3, 20)),
    'min_samples_split': np.arange(2, 20),
    'min_samples_leaf': np.arange(1, 20),
}

# Búsqueda aleatorizada para el Árbol de Decisión
tree = DecisionTreeClassifier()
tree_search = RandomizedSearchCV(tree, param_distributions=param_dist_tree,
                                n_iter=100, cv=5, verbose=1, n_jobs=-1)
tree_search.fit(X_train_filtered, y_train_filtered)

print("Mejores hiperparámetros para Árbol de Decisión:", tree_search.best_params_)

# Hiperparámetros para el Bosque Aleatorio
param_dist_rf = {
    'n_estimators': np.arange(10, 201),
    'max_depth': [None] + list(np.arange(3, 20)),
    'min_samples_split': np.arange(2, 20),
    'min_samples_leaf': np.arange(1, 20),
    'bootstrap': [True, False]
}

# Búsqueda aleatorizada para el Bosque Aleatorio
rf = RandomForestClassifier()
rf_search = RandomizedSearchCV(rf, param_distributions=param_dist_rf,
                              n_iter=100, cv=5, verbose=1, n_jobs=-1)
rf_search.fit(X_train_filtered, y_train_filtered)

print("Mejores hiperparámetros para Bosque Aleatorio:", rf_search.best_params_)
```

Fitting 5 folds for each of 100 candidates, totalling 500 fits

Mejores hiperparámetros para Árbol de Decisión: {'splitter': 'random', 'min_samples_split': 10, 'min_samples_leaf': 6, 'max_depth': 17, 'criterion': 'gini'}

Fitting 5 folds for each of 100 candidates, totalling 500 fits

Mejores hiperparámetros para Bosque Aleatorio: {'n_estimators': 100, 'min_samples_split': 10, 'min_samples_leaf': 2, 'max_depth': 17, 'bootstrap': True}

```

In [65]: from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

# Divide los datos en características (X) y etiquetas (y) después de filtrar los datos
X_filtered = data_filtered[['Tenure in Months', 'Monthly Charge', 'Total Customer Service Calls']]
y_filtered = data_filtered['Churn Value']

# Divide los datos en conjuntos de entrenamiento y prueba
X_train_filtered, X_test_filtered, y_train_filtered, y_test_filtered = train_test_split(X_filtered, y_filtered,
                                                                                          test_size=0.3, random_state=42)

# Árbol de Decisión con hiperparámetros optimizados
tree_model_filtered = DecisionTreeClassifier(
    splitter='random',
    min_samples_split=10,
    min_samples_leaf=6,
    max_depth=17,
    criterion='gini'
)

tree_model_filtered.fit(X_train_filtered, y_train_filtered)
y_pred_tree_filtered = tree_model_filtered.predict(X_test_filtered)

print("Métricas para Árbol de Decisión con datos filtrados y hiperparámetros optimizados:")
print("Precisión:", precision_score(y_test_filtered, y_pred_tree_filtered))
print("Recall:", recall_score(y_test_filtered, y_pred_tree_filtered))
print("F1-Score:", f1_score(y_test_filtered, y_pred_tree_filtered))
print("Área bajo la curva ROC:", roc_auc_score(y_test_filtered, y_pred_tree_filtered))
print("-----")

# Bosque Aleatorio con hiperparámetros optimizados
rf_model_filtered = RandomForestClassifier(
    n_estimators=100,
    min_samples_split=10,
    min_samples_leaf=2,
    max_depth=17,
    bootstrap=True
)

rf_model_filtered.fit(X_train_filtered, y_train_filtered)
y_pred_rf_filtered = rf_model_filtered.predict(X_test_filtered)

print("Métricas para Bosque Aleatorio con datos filtrados y hiperparámetros optimizados:")
print("Precisión:", precision_score(y_test_filtered, y_pred_rf_filtered))
print("Recall:", recall_score(y_test_filtered, y_pred_rf_filtered))
print("F1-Score:", f1_score(y_test_filtered, y_pred_rf_filtered))
print("Área bajo la curva ROC:", roc_auc_score(y_test_filtered, y_pred_rf_filtered))

```

Métricas para Árbol de Decisión con datos filtrados y hiperparámetros optimizados:

Precisión: 0.9375

Recall: 0.9022556390977443

F1-Score: 0.9195402298850575

Área bajo la curva ROC: 0.9340338024548551

Métricas para Bosque Aleatorio con datos filtrados y hiperparámetros optimizados:

Precisión: 0.9453125

Recall: 0.9097744360902256

F1-Score: 0.9272030651340997

Área bajo la curva ROC: 0.9399299530878479

Conclusión: Ambos modelos, el Árbol de Decisión y el Bosque Aleatorio, han mostrado un rendimiento sólido con los datos filtrados y los hiperparámetros optimizados. Si comparamos estas métricas con las obtenidas anteriormente, vemos que la optimización de hiperparámetros ha mejorado el rendimiento de los modelos, especialmente en términos de precisión.

El Bosque Aleatorio tiene un ligero borde sobre el Árbol de Decisión en términos de todas las métricas. Por lo tanto, si tuviera que elegir entre los dos modelos para una implementación en un entorno de producción, el Bosque Aleatorio con hiperparámetros optimizados sería la elección recomendada debido a su rendimiento superior en estos datos. Sin embargo, es importante considerar otros factores, como la interpretabilidad y la eficiencia computacional, al tomar decisiones finales sobre el modelo a utilizar.

Conclusion modelos de clasificacion:

La opción 2, que consistió en eliminar los valores de "Customer Satisfaction" iguales a 0, resultó en modelos de clasificación más precisos en comparación con la opción 1, que agregó una categoría adicional para los clientes que no contestaron la encuesta. La limpieza de datos en la opción 2 mejoró significativamente el rendimiento de los modelos, lo que sugiere que eliminar valores faltantes es clave para obtener resultados más precisos en la predicción del churn.

Por ende se recomienda implementar estrategias para alentar a los clientes a contestar la encuesta de satisfacción. Algunas posibles recomendaciones son:

Incentivos: Ofrecer incentivos a los clientes para que completen la encuesta, como descuentos, recompensas o sorteos de premios.

Comunicación clara: Explicar la importancia de la encuesta y cómo sus comentarios ayudarán a mejorar los servicios y productos ofrecidos.

Facilidad de acceso: Asegurarse de que la encuesta sea fácil de acceder y completar, preferiblemente en línea o mediante una aplicación móvil.

Recordatorios: Enviar recordatorios periódicos a los clientes para que completen la encuesta, sin ser demasiado intrusivos.

Feedback transparente: Demostrar que los comentarios de los clientes son valorados y que se están tomando acciones en base a sus respuestas.

Personalización: Adaptar las encuestas a las preferencias y necesidades específicas de cada cliente para aumentar la relevancia y la probabilidad de respuesta.

Seguimiento: Realizar un seguimiento con los clientes que no han contestado la encuesta para entender las razones detrás de su falta de respuesta y buscar soluciones para mejorar la tasa de participación.

Al implementar estas estrategias, es probable que se aumente la tasa de respuesta de los clientes a la encuesta de satisfacción, lo que proporcionará una mayor cantidad de datos para mejorar la toma de decisiones y el rendimiento general del negocio.

In []:

In []: