Predicción de Abandono de Clientes en Telecomunicaciones

Andrés Felipe Camargo Pinilla

David Santiago Gaona Pardo

Juan David Moreno Suárez

Santiago Sabogal Millan

Tomás Barón Galvis

1. Resumen

Este informe presenta un análisis detallado sobre el abandono de clientes en telecomunicaciones, a través de distintos modelos predictivos. Utilizando un conjunto de datos unificado que contiene información tanto demográfica como del uso de servicios, se realizará una exploración estadística para identificar patrones clave y plantear tres modelos predictivos de distinto tipo. Además, se propondrá un plan de retención basado en los hallazgos que ayude a mitigar el abandono de clientes.

2. Introducción

2.1. Explicación del problema

En las empresas de servicios, existe una métrica crucial para identificar problemas en la calidad del servicio y los números a futuro, la tasa de abandono o también llamada churn rate. En la industria de telecomunicaciones, la tasa de abandono es una métrica crítica que impacta directamente la rentabilidad, debido a que involucra que tan probable es que un cliente decida suspender su relación con la compañía y cambiarse a otra. Según un reporte hecho por McKinsey & Company (2023) acerca de la tasa de abandono en empresas de telecomunicaciones a nivel global, la industria pierde $65 billones anuales solamente por abandono de clientes. Una forma de contrarrestar el fenómeno del abandono suele ser la priorización por nuevos clientes con campañas y promociones, después de todo, dentro del esquema capitalista de la competencia, las empresas buscan ofrecer algo que destaque y llame la atención, sin embargo según el mismo reporte el costo de adquirir nuevos clientes es 5 a 7 veces mayor que retener existentes. Los clientes existentes son la prioridad de cada empresa de telecomunicaciones y mantenerlos es su deber.

2.2. Objetivo general

Analizar la tasa de abandono de clientes de una empresa de telecomunicaciones a través del planteamiento de tres modelos predictivos, identificando los factores que crean tendencias a mantenerse en la compañía o abandonarla, para de esta manera, proponer un plan de retención que ayude a mitigar el abandono de clientes. Según el reporte de McKinsey (2023), lograr reducir el churn en un 5 % puede aumentar ganancias hasta un 25 %

2.3. Objetivos específicos

- Implementar tres modelos clásicos de ML (Perceptrón, Adaline, Regresión Logística) para predecir la tasa de abandono.
- Evaluar su desempeño en condiciones realistas (datos desbalanceados).
- Identificar los factores de mayor impacto en la retención
- Proponer estrategias accionables basadas en insights del modelo.

3. Metodología

3.1. Fuente de datos

El procesamiento de datos se llevó acabo utilizando las siguientes librerías importadas en el notebook trabajado (Nota: Cada "snippet" de código que requiera de una librería extra será mencionado en su apartado)

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.model_selection import train_test_split
```

Los datos utilizados en este estudio fueron obtenidos de un archivo CSV denominado Telco-Customer-Churn-V2.csv, el cual fue alojado en un repositorio de GitHub y accedido a través de su enlace. La importación de los datos se realizó mediante Python utilizando la biblioteca de pandas. A continuación, se muestra el fragmento de código utilizado:

```
url = "https://raw.githubusercontent.com/Tommybg/Proyecto2_Analitica/refs/heads/main/Telco-Customer-Churn-V2.csv"

df = pd.read_csv(url)
df.head()
```

El dataset contiene información detallada sobre los clientes afiliados o alguna vez afiliados a la empresa, incluyendo variables demográficas, contratos, uso de servicios y monetarias.

3.2. Preprocesamiento de datos

3.2.1. Importar los datos en Python

El dataset fue cargado en un DataFrame de pandas para su posterior procesamiento y análisis.

3.2.2. Identificar y manejar valores nulos o inconsistentes

Se realizó una inspección inicial de los datos para identificar valores nulos y atípicos. En casos donde se encontraron datos faltantes fueron tratados según la proporción faltante; Si supera el 30% serán eliminados como registros.

```
# Calculamos el porcentaje que equivalen los datos faltantes
missing_percentage = df.isnull().mean() * 100

# Eliminamos las columnas que tengan más del 30 porciento de datos
faltantes
columns_to_drop = missing_percentage[missing_percentage >
30].index.tolist()
df = df.drop(columns=columns_to_drop)

print("Percentage of missing values in each column:")
print(missing_percentage.to_markdown(numalign="left", stralign="left"))
print(f"Columns dropped: {columns_to_drop}")
```

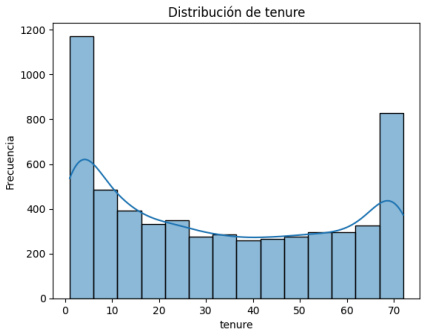
Los datos faltantes restantes que no ameritaba su eliminación se trataron con métodos de imputación. Los campos de Dependents, tenure, OnlineSecuirty, OnlineBackup, TechSupport, MonthlyCharges y TotalCharges que aún presentaban datos faltantes se les aplicó imputación, para las variables cuantitativas, el método de imputación seleccionado fue determinado por la distribución normal de las variables en el conjunto; una distribución normal se trata con imputación de media mientras que una distribución que no es normal se imputa por mediana.

Figura 1.1.0

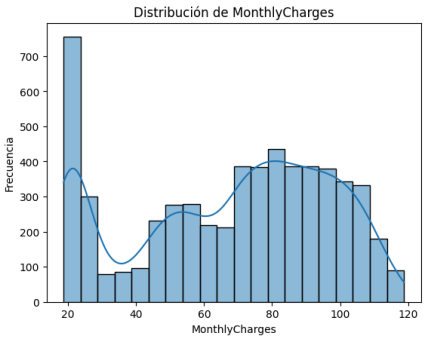
(Distribución tenure)

Figura 1.2.0

(Distribución MonthlyCharges)



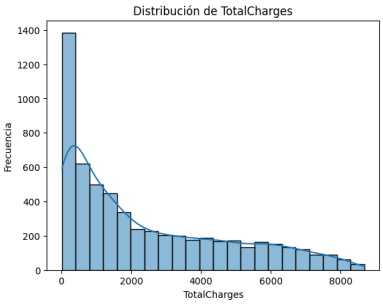
statistic = 0.8944593298276179, p-value = 0.0



statistic = 1.0, p-value = 0.0

Figura 1.3.0

(Distribución TotalCharges)



statistic = 1.0, p-value = 0.0

Después de analizar las distribuciones de cada variable y determinar con el KS test que ninguna de ellas sigue una distribución normal se opta por imputar con la mediana generando los siguientes resultados:

```
# Imputar los valores faltantes con la mediana para las columnas
especificadas
for col in ['tenure', 'MonthlyCharges', 'TotalCharges']:
    median_val = df[col].median()
    print(f"Imputando valores faltantes en {col} con la mediana:
{median_val}")
    df[col] = df[col].fillna(median_val)
```

```
# Verificar si aún existen valores faltantes
missing_values_after = df.isnull().sum()
print("\nValores faltantes después de la imputación:")
print(missing_values_after)

# Calcular las estadísticas descriptivas después de la imputación
desc_stats_after = df.describe()

# Comparar las estadísticas descriptivas antes y después de la
imputación
print("\nComparación de estadísticas descriptivas (Antes - Después):")
comparison_df = desc_stats_before - desc_stats_after
print(comparison_df)
```

Figura 1.1.1

(Distribución tenure comparativa)

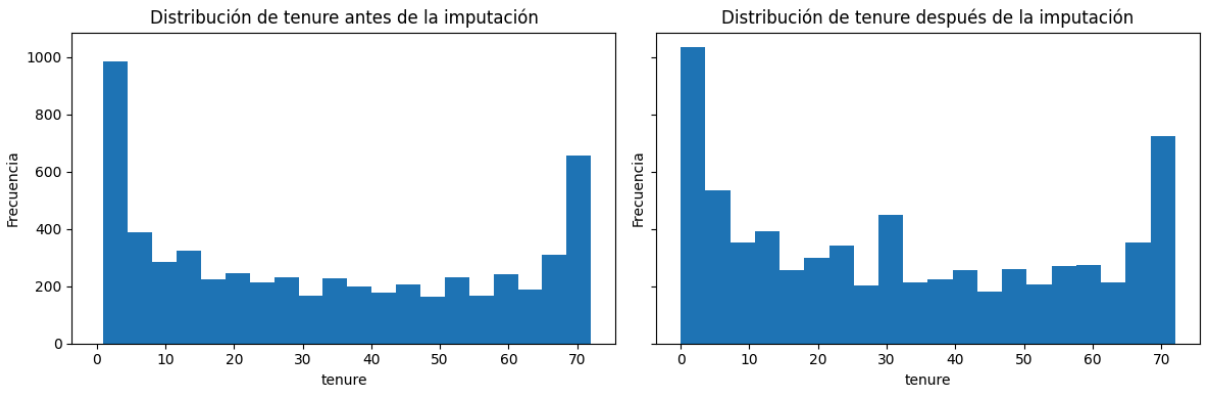


Figura 1.2.1

(Distribución MonthlyChanges comparativa)

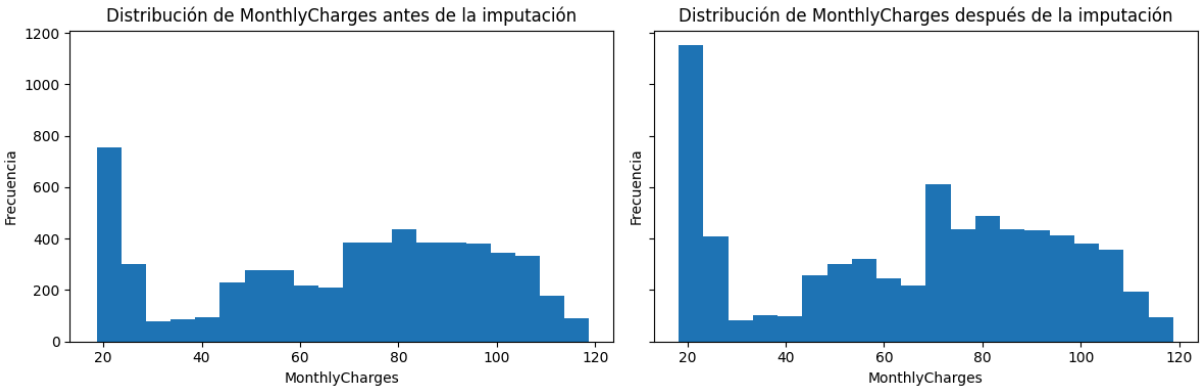
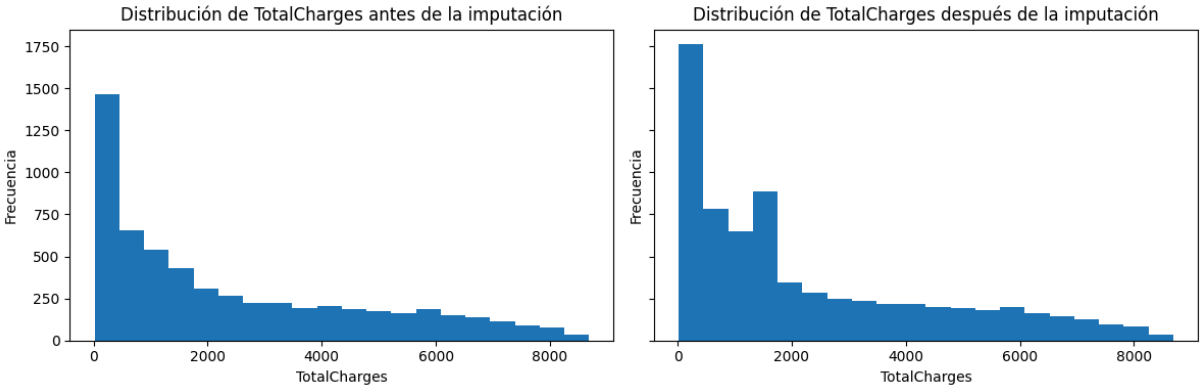


Figura 1.3.1

(Distribución TotalChanges comparativa)



Respecto a los datos faltantes para las variables cualitativas, se optó por reemplazar por la moda, debido al porcentaje tan bajo que estas implican. Primero, se verificó cada moda.

Figura 1.1.0

(Distribución Dependents)

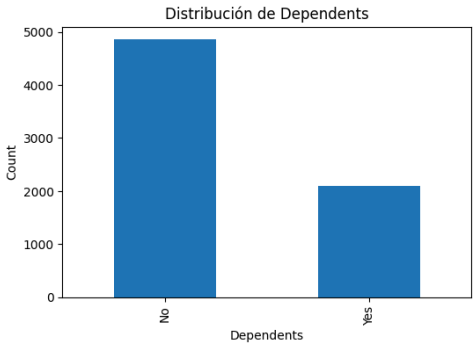


Figura 1.2.0

(Distribución TechSupport)

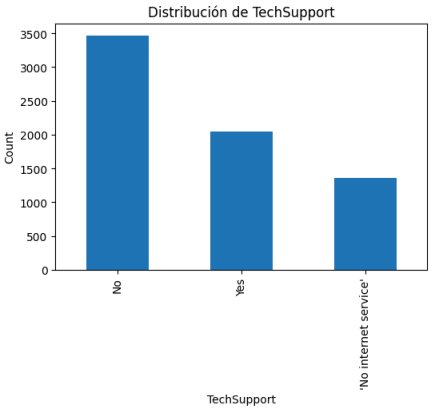


Figura 1.3.0

(Distribución OnlineSecurity)

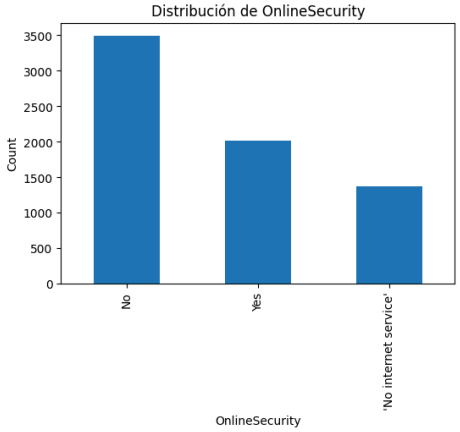
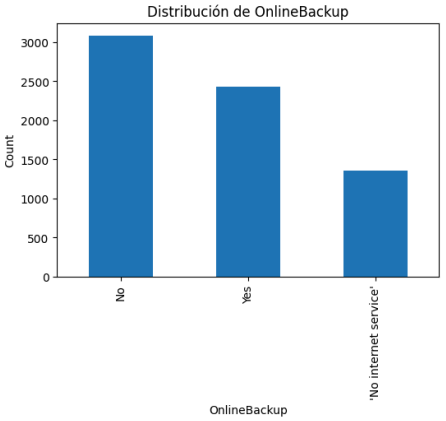


Figura 1.4.0

(Distribución OnlineBackup)



Es destacable que todas las modas en cada una de las variables son la respuesta ‘No’. Sabiendo eso, se inicia el proceso de imputación.

```
# Imputar valores faltantes con la moda para las columnas especificadas
cols_to_impute = ['Dependents', 'TechSupport', 'OnlineSecurity',
'OnlineBackup']
for col in cols_to_impute:
    # Verificar si la columna tiene valores faltantes
    if df[col].isnull().any():
        # Obtener la moda de la columna
        mode_val = df[col].mode()[0]

        # Imputar los valores faltantes con la moda
        print(f"Imputando valores faltantes en {col} con la moda:
{mode_val}")
        df[col] = df[col].fillna(mode_val)
    else:
        print(f"La columna {col} no tiene valores faltantes.")

# Verificar si aún existen valores faltantes DESPUÉS de la imputación
con la moda
```

```
missing_values_after_mode = df.isnull().sum()
print("\nValores faltantes después de la imputación con la moda:")
print(missing_values_after_mode)
```

3.2.3. Datos atípicos

El paso a seguir es el tratamiento de datos atípicos que influyen significativamente en los datos. Para identificarlos se graficaron box-plots para todas las variables numéricas implementando la estrategia adecuada para cada variable. Se tendrá en cuenta la cantidad de atípicos, proporción y características de la variable numérica.

Figura 1.1.0

(tenure)

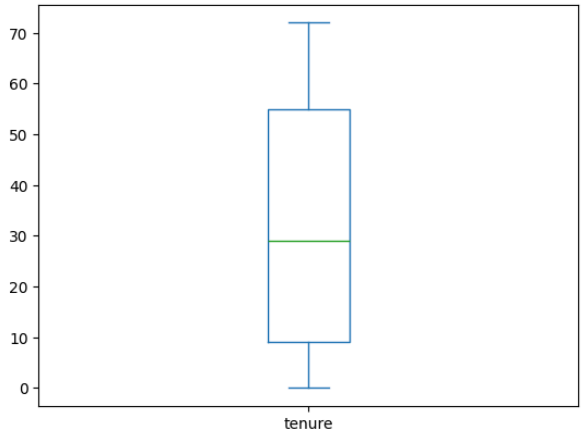


Figura 1.2.0

(MonthlyCharges)

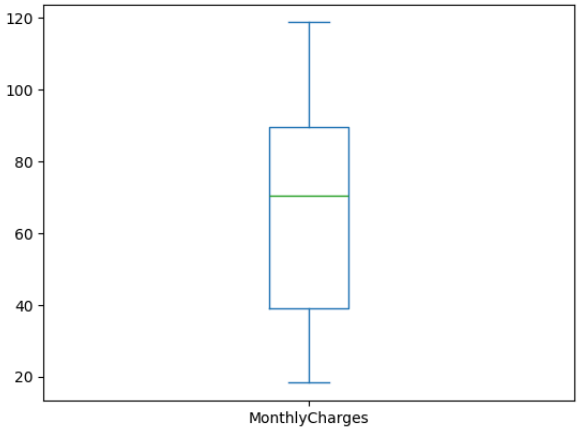


Figura 1.3.0

(TotalCharges)

Se identifica que la única variable con datos atípicos es TotalCharges. Después de haber aplicado Winsorización y reemplazamiento con media truncada en segundo plano, no se obtuvieron buenos resultados. Por ende, la tercera vía era reemplazar los outliers con el valor permitido más cercano, el cual está entre los percentiles 5% y 95%. A pesar de alterar la variabilidad al incorporar datos más dispersos, esta fue la única opción funcional para tratar los outliers manteniendo la totalidad de los datos.

```
# Se imputan los datos con los percentiles 5 y 95
def imputar_outliers_percentiles(df, columna):
    p5 = df[columna].quantile(0.05)
    p95 = df[columna].quantile(0.95)

    df[columna] = np.clip(df[columna], p5, p95)

columnasFinales = ["TotalCharges"]

for col in columnasFinales:
    imputar_outliers_percentiles(df, col)
```

3.3. Tratamiento de los datos

3.3.1. Imputación de la variable TotalCharges

Para empezar con el tratamiento de los datos, se solicita que la variable TotalCharges sea la imputación de la multiplicación entre la variable tenure y MonthlyCharges para asegurar su

coherencia. Tenure son los meses que el cliente estuvo con el servicio y MonthlyCharges es el cobro mensual que se le realizó, por lo que el total (TotalCharges) es la multiplicación de esos dos.

```
df['TotalCharges'] = df['tenure'] * df['MonthlyCharges']
```

3.3.2. One-hot encoding

Se solicita que las variables del tipo Demográficas como gender, SeniorCitizen, Partner y Dependents sean tratadas con el método One-Hot Encoding. El One-Hot Encoding es una técnica de preprocesamiento utilizada para convertir variables categóricas en una representación numérica binaria. Cada categoría se convierte en una columna independiente con valores 0 o 1, indicando la presencia de esa categoría en una muestra.

```
# Aplicar One-Hot Encoding para tipo Demográficas
df = pd.get_dummies(df, columns=['gender', 'SeniorCitizen','Partner',
'Dependents'], drop_first=True)
```

3.3.3. Ordinal encoding

Prosiguiendo, se solicita que las variables del tipo Contrato como Contract, PaperlessBilling y PaymentMethod sean tratadas con el método de Ordinal Encoding. El Ordinal Encoding es una técnica de preprocesamiento utilizada para convertir variables categóricas en valores numéricos de manera ordenada. A cada categoría se le asigna un número entero en función de un orden lógico.

```
# Definir las columnas a transformar
contract_columns = ['Contract', 'PaperlessBilling', 'PaymentMethod']

# Definir el orden de las categorías en cada columna
contract_order = ["Month-to-month", "'One year'", "'Two year'"]
payment_method_order = ["'Electronic check'", "'Mailed check'", "'Bank
transfer (automatic)'", "'Credit card (automatic)'"]

# Crear el codificador
encoder = OrdinalEncoder(categories=[contract_order, ['No', 'Yes'],
```

```
payment_method_order])

# Aplicar el encoding solo a las columnas específicas
df[contract_columns] = encoder.fit_transform(df[contract_columns])

# Convertir los valores a enteros
df[contract_columns] = df[contract_columns].astype(int)
```

3.3.4. Binary encoding

A continuación, se solicita que las variables del tipo Uso de servicios sean tratadas con el método binario. Debido a que todas estas incluyen una categoría extras, simplemente se clasificará como 0 o 1, dependiendo el contexto.

```
service_columns = [
    'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
    'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
'StreamingMovies'
]

# Aplicar la conversión a binario
df[service_columns] = df[service_columns].replace({'Yes': 1, 'DSL':1,
"'Fiber optic'":1, 'No': 0, "'No internet service'": 0, "'No phone
service'": 0})
```

3.3.5. Escalar

Respecto al tratamiento escalar, el cual se solicita para las variables de tipo monetarias como tenure, MonthlyCharges y TotalCharges, no es necesario tratamiento, ya que las variables ya están escaladas al ser numéricas.

4. Análisis Exploratorio

Como es bien sabido el análisis exploratorio de datos (EDA) es un paso crucial para cualquier proyecto de análisis de datos, debido a que este permite obtener una comprensión un poco más profunda del conjunto de datos antes de pasar a realizar un análisis más complejo. El objetivo de este EDA es lograr identificar patrones, tendencias y relaciones entre las variables, lo que proporciona una

gran cantidad de información valiosa para pasar a la toma de decisiones. En este análisis exploratorio, se utilizaron técnicas como el cálculo de estadísticas descriptivas, la implementación de matrices de correlación, implementación de tablas de contingencia y la visualización de datos a través de gráficos. Esta parte del informe comenzará con el análisis de distribuciones de las variables. Seguido del cálculo de estadísticas descriptivas. Luego, se examinarán las correlaciones entre las variables y se presentarán visualizaciones para una mejor comprensión.

4.1. Estadísticas descriptivas

Se calcularon estadísticas descriptivas, como la media, mediana, desviación estándar, valores mínimo y máximo, para resumir las características principales de las variables. Este análisis detallado solo se pudo realizar para las tres variables cuantitativas.

Estadísticas Ahora:

	tenure	MonthlyCharges	TotalCharges
count	7043.0	7043.000000	7043.0
mean	32.227034	64.943149	2267.517989
std	24.224531	29.716655	2231.349316
min	0.0	18.250000	0.0
25%	9.0	39.100000	418.4
50%	29.0	70.300000	1422.4
75%	55.0	89.600000	3672.9
max	72.0	118.750000	8550.0

	tenure	MonthlyCharges	TotalCharges
count	7043	7043	7043
mean	32.227	64.9431	2267.52
std	24.2245	29.7167	2231.35
min	0	18.25	0
25%	9	39.1	418.4
50%	29	70.3	1422.4
75%	55	89.6	3672.9
max	72	118.75	8550

Figura 4.1.1

(Estadísticas descriptivas)

4.1.1. Matriz de correlación

Esta gráfica de correlación muestra las relaciones lineales existentes entre variables del conjunto de datos. Para entender los valores que nos da vamos a explicar un poco su valor y que representa, la correlación se mide en un rango de -1 a 1 donde -1 significa que la correlación es

negativa perfecta, 1 es correlación positiva perfecta y 0 indica que no existe correlación lineal, teniendo esto en cuenta.

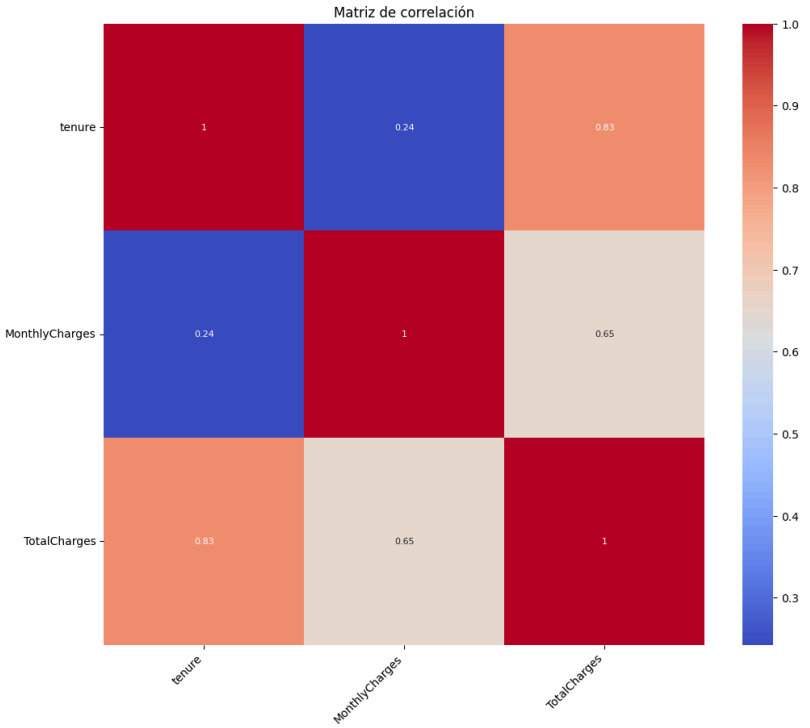


Figura 4.1.1.1

(Matriz de correlación)

4.1.2. Tablas de contingencia

Una tabla de contingencia es una tabla que muestra la relación entre dos o más variables categóricas. Permite analizar cómo se distribuyen los valores de una variable en función de otra.

```
Tabla de Contingencia: gender vs Churn
Churn          No       Yes
gender
Female   0.730791  0.269209
Male     0.738397  0.261603
```

```
--------------------------------------------------

Tabla de Contingencia: SeniorCitizen vs Churn
Churn                No       Yes
SeniorCitizen
No             0.763938  0.236062
Yes            0.583187  0.416813

--------------------------------------------------

Tabla de Contingencia: Partner vs Churn
Churn          No       Yes
Partner
No       0.670420  0.329580
Yes      0.803351  0.196649

--------------------------------------------------

Tabla de Contingencia: Dependents vs Churn
Churn             No       Yes
Dependents
No          0.687234  0.312766
Yes         0.846190  0.153810

--------------------------------------------------

Tabla de Contingencia: PhoneService vs Churn
Churn               No       Yes
PhoneService
No            0.750733  0.249267
Yes           0.732904  0.267096

--------------------------------------------------
```

```
Tabla de Contingencia: MultipleLines vs Churn
Churn                     No       Yes
MultipleLines
'No phone service'  0.750733  0.249267
No                  0.749558  0.250442
Yes                 0.713901  0.286099

--------------------------------------------------

Tabla de Contingencia: InternetService vs Churn
Churn                  No       Yes
InternetService
'Fiber optic'    0.581072  0.418928
DSL              0.810409  0.189591
No               0.925950  0.074050

--------------------------------------------------

Tabla de Contingencia: OnlineSecurity vs Churn
Churn                        No       Yes
OnlineSecurity
'No internet service'  0.929619  0.070381
No                     0.596175  0.403825
Yes                    0.853888  0.146112

--------------------------------------------------

Tabla de Contingencia: OnlineBackup vs Churn
Churn                        No       Yes
OnlineBackup
'No internet service'  0.927888  0.072112
No                     0.616590  0.383410
Yes                    0.784685  0.215315

--------------------------------------------------
```

```
Tabla de Contingencia: DeviceProtection vs Churn
Churn                         No       Yes
DeviceProtection
'No internet service'  0.925950  0.074050
No                     0.608724  0.391276
Yes                    0.774979  0.225021

--------------------------------------------------

Tabla de Contingencia: TechSupport vs Churn
Churn                         No       Yes
TechSupport
'No internet service'  0.923021  0.076979
No                     0.600000  0.400000
Yes                    0.848337  0.151663

--------------------------------------------------

Tabla de Contingencia: StreamingTV vs Churn
Churn                         No       Yes
StreamingTV
'No internet service'  0.925950  0.074050
No                     0.664769  0.335231
Yes                    0.699298  0.300702

--------------------------------------------------

Tabla de Contingencia: StreamingMovies vs Churn
Churn                         No       Yes
StreamingMovies
'No internet service'  0.925950  0.074050
No                     0.663196  0.336804
Yes                    0.700586  0.299414
```

```
------------------------------------------------

Tabla de Contingencia: Contract vs Churn
Churn                 No       Yes
Contract
'One year'      0.887305  0.112695
'Two year'      0.971681  0.028319
Month-to-month  0.572903  0.427097

------------------------------------------------

Tabla de Contingencia: PaperlessBilling vs Churn
Churn                   No       Yes
PaperlessBilling
No                0.836699  0.163301
Yes               0.664349  0.335651

------------------------------------------------

Tabla de Contingencia: PaymentMethod vs Churn
Churn                               No       Yes
PaymentMethod
'Bank transfer (automatic)'  0.832902  0.167098
'Credit card (automatic)'    0.847569  0.152431
'Electronic check'           0.547146  0.452854
'Mailed check'               0.808933  0.191067

------------------------------------------------
```

4.2. Distribución de variables categóricas vs Churn

Para esta sección, se realizaron gráficos de barras de la distribución de las distintas variables categóricas según Churn.

Figura 3.3.4.1

(gender)

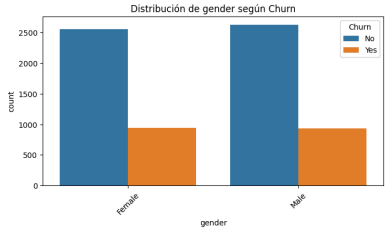


Figura 3.3.4.2

(SeniorCitizen)

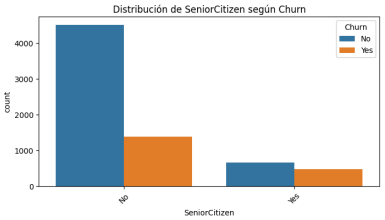


Figura 3.3.4.3

(Partner)

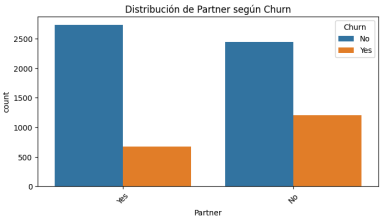


Figura 3.3.4.4

(Dependent)

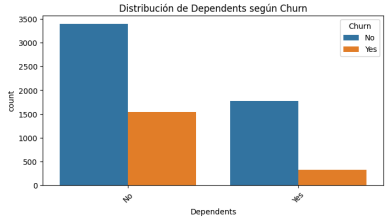


Figura 3.3.4.5

(PhoneSevice)

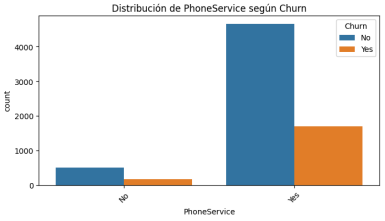


Figura 3.3.4.6

(MultipleLines)

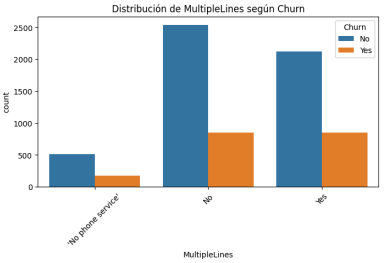


Figura 3.3.4.7

(InternetService)

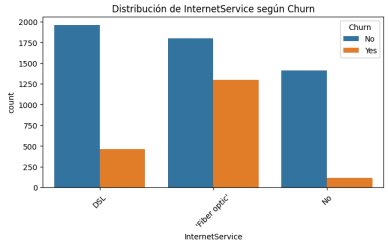


Figura 3.3.4.8

(OnlineSecurity)

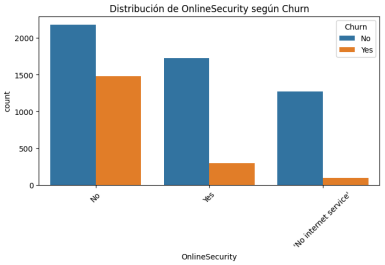


Figura 3.3.4.9

(OnlineBackup)

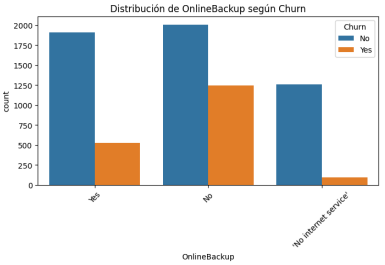


Figura 3.3.4.1

(DeviceProtection)

Figura 3.3.4.2

(TechSupport)

Figura 3.3.4.3

(StreamingTV)

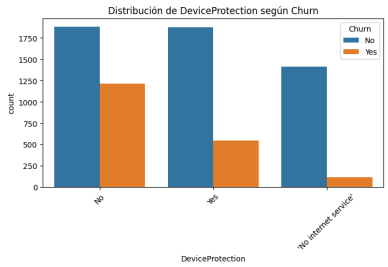


Figura 3.3.4.4

(StreamingMovies)

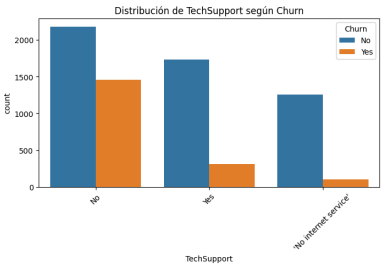


Figura 3.3.4.5

(Contract)

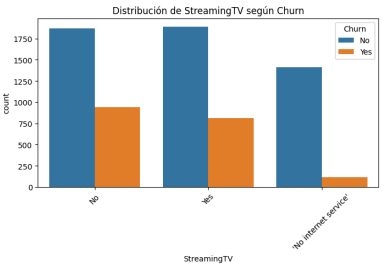
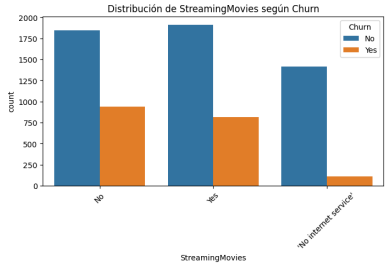
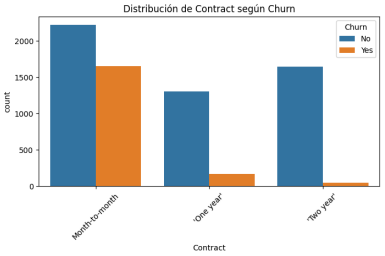


Figura 3.3.4.6

(PaperlessBilling)





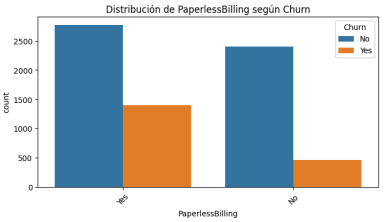
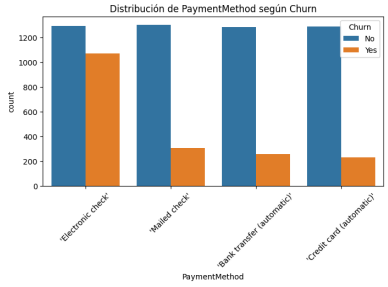


Figura 3.3.4.7

(PaymentMethod)



4.3. Distribución de variables numéricas vs Churn

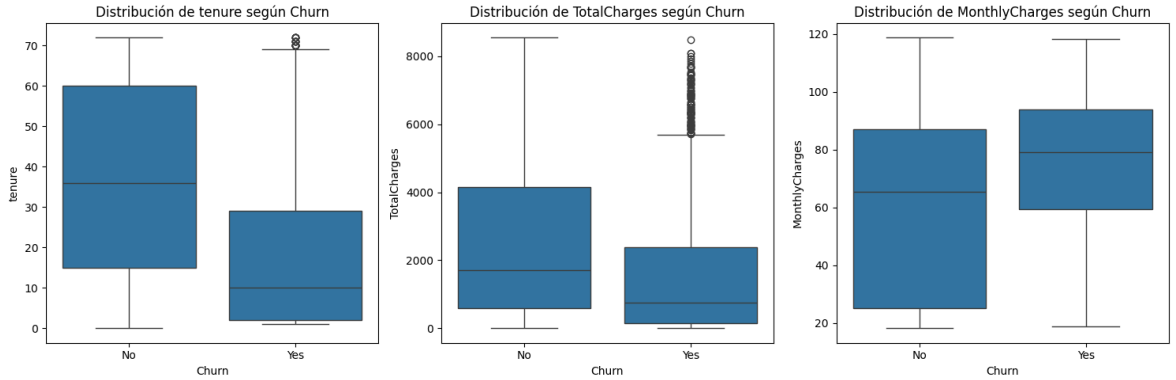
Para esta sección, se realizaron dos tipos de gráficos para la distribución de las variables numéricas según Churn, el boxplot y el violín plot.

4.3.1. Boxplot

Los boxplots muestran la distribución de valores numéricos dentro de cada categoría de Churn.

Figura 3.3.4.0.2

(Boxplot de variables numéricas según Churn)



Se interpreta de la siguiente manera: Si la mediana es diferente entre *Yes* y *No*, significa que la variable numérica influye en el Churn.

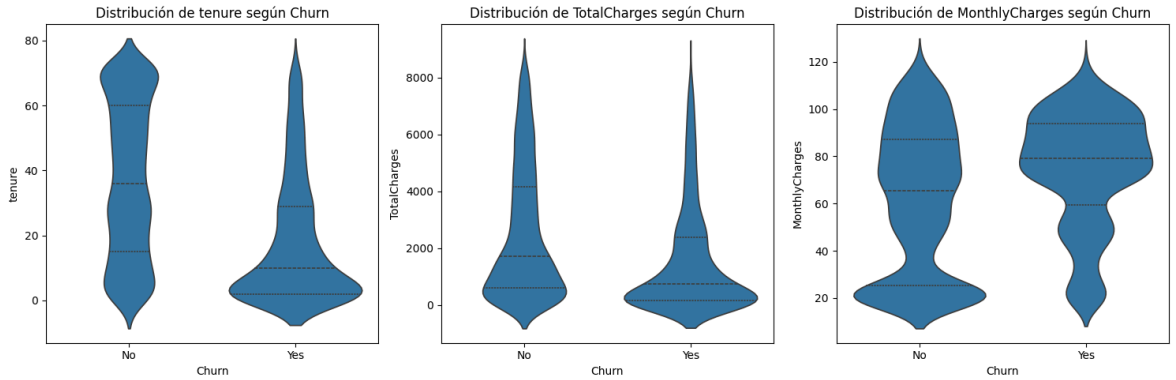
Es evidenciable que las medianas dentro de cada una de las tres gráficas son diferentes, por ende es posible concluir que las variables numéricas tienen cierta influencia en Churn.

4.3.2. Violinplot

Los violinplots combinan el boxplot con una estimación de densidad, mostrando la distribución completa de los datos.

Figura 3.3.4.0.2

(Violinplot de variables numéricas según Churn)



Su interpretación es: Si la forma de las distribuciones entre *Yes* y *No* es muy distinta, la variable podría ser un buen predictor de Churn, mientras que los picos altos indican que muchos valores se concentran en ciertos rangos.

Es evidenciable que la forma de los gráficos dentro de cada uno de los tres gráficos es significativamente distinta, por ende es posible declarar que las variables numéricas pueden ser predictores para Churn.

4.4. Distribución Tenure vs Churn

Respecto al análisis anterior, es posible concluir que la mejor forma de entender el abandono de la gente es a través de tenure. Replicando el diagrama boxplot previo, es posible concluir que, además de tener cierta influencia en Churn, **los clientes con menor tiempo de permanencia tienden a irse y viceversa**.

4.5. Tasa de abandono por categoría de servicio

Para determinar una tasa de abandono por servicio, se planteó la siguiente fórmula:

$$ChurnRate_{Ci} = \left(\frac{\sum_{j=1}^{N_i} Churn_j}{N_i}\right) \times 100$$

- C_i representa una categoría específica.
- N_i es el número total de clientes en la categoría C_i.
- $Churn_j$ toma el valor 1 si el cliente abandonó y 0 si se quedó.
- $\sum_{j=1}^{N_i} Churn_j$ representa la cantidad de clientes que abandonaron en esa categoría.
- Multiplicamos por 100 para expresarlo en porcentaje.

5. Modelo Perceptrón

5.1. Arquitectura del Modelo

El Perceptrón es un modelo de aprendizaje supervisado que aplica una combinación lineal de características de entrada seguida de una función de activación escalonada para tomar decisiones binarias.

5.2. Preprocesamiento de datos

División de datos: Se separan los datos en un 80% de entrenamiento y 20% de prueba, asegurando estratificación para mantener la proporción de clases.

```
# Separar características (X) y variable objetivo (y)

X = df.drop(columns=["Churn"])

y = df["Churn"] # Esta es la variable objetivo: 1 (abandono) o 0
(retención)



# Dividir los datos en conjuntos de entrenamiento y prueba (80%-20%)

# Esto es importante porque necesitamos evaluar el modelo en datos que no
ha visto antes

X_train, X_test, y_train, y_test = train_test_split(

  X, y, test_size=0.2, random_state=42, stratify=y

)
```

Normalización de características: Se usa StandardScaler para garantizar que todas las variables contribuyan de manera equitativa al modelo.

```
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)  # Ajustamos el escalador
en los datos de entrenamiento
```

```
X_test_scaled = scaler.transform(X_test)  # Usamos la misma transformación
en los datos de prueba
```

5.3. Optimización de Hiperparámetros

Se probó una serie de combinaciones de hiperparámetros para determinar la mejor configuración del modelo.

Hiperparámetro	Valores Evaluados	Mejor Valor
alpha	[0.0001, 0.001, 0.01, 0.1, 1]	0.0001
penalty	['l1', 'l2', None]	l1
max_iter	1000	1000
tol	1e-3	1e-3

En total se realizaron 15 combinaciones, donde se variaron estos hiperparámetros hasta encontrar el mejor modelo. El criterio para escoger el mejor modelo fue comparándolos hasta encontrar el modelo con mejor precisión.

5.4. Interpretación de los Hiperparámetros Óptimos

El mejor modelo encontrado utiliza:

- **Penalty = L1 (Lasso):** Esto significa que el modelo aplica una regularización L1, que puede hacer que algunos coeficientes se reduzcan a cero, seleccionando solo las características más relevantes y promoviendo la esparsidad en los pesos del modelo.

- **Alpha = 0.0001:** Un valor pequeño de alpha indica una **regularización mínima**, lo que permite que el modelo aprenda más de los datos sin ser demasiado restrictivo.

Esta combinación ha resultado en el mejor desempeño en términos de **precisión global (77.5%)**.

5.5. Consideraciones sobre la Convergencia

El modelo ha sido entrenado con un límite de **1000 iteraciones** y una tolerancia (tol=1e-3) que define el criterio de parada. Dado que la regularización L1 tiende a generar pesos escasos y puede afectar la estabilidad del proceso de optimización, es posible que un número mayor de iteraciones o una tolerancia más baja puedan mejorar la convergencia.

Además, el uso de random_state=42 asegura que los resultados sean reproducibles.

5.6. Evaluación de Métricas

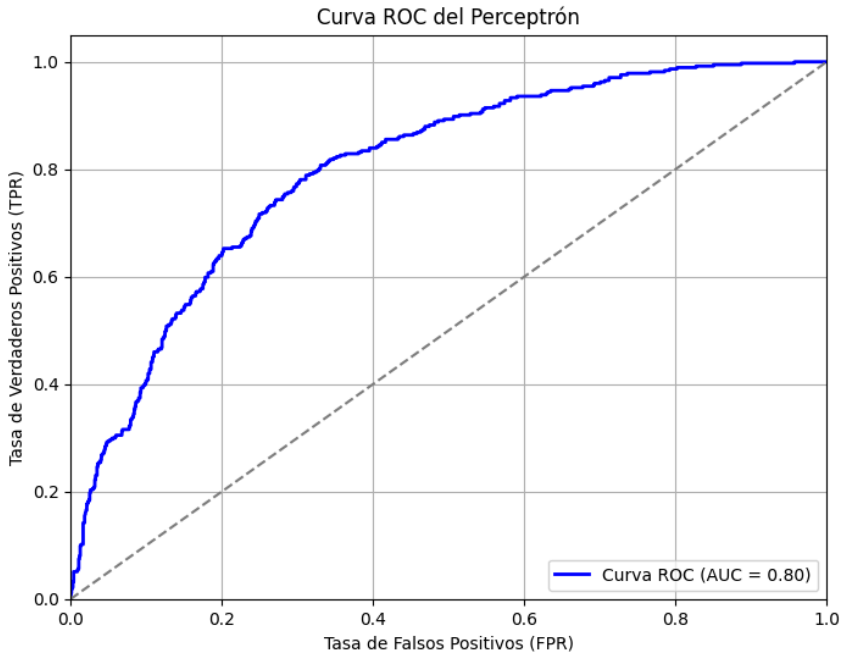
Métrica	Valor	Interpretación
Precisión	0.7750	El modelo clasifica correctamente el 77.5% de los casos, pero esta métrica no es suficiente si las clases están desbalanceadas.
Recall (Churn=Yes)	0.2513	El modelo solo detecta el 25.1% de los clientes que realmente abandonan, lo que indica que tiene dificultades para identificar churn.
F1-Score	0.3723	Dado el bajo recall, el F1-Score también es bajo (37.2%), lo que sugiere un desequilibrio entre precisión y recall.
AUC-ROC	0.8005	Este es un buen valor (80%), lo que indica que el modelo es capaz de distinguir entre clientes que se quedan y los que se van con un rendimiento aceptable.

5.7. Interpretación de las Gráficas

Curva ROC

- Se observa una curva ROC con un área bajo la curva (AUC) de 0.80, lo que indica una buena capacidad de discriminación.

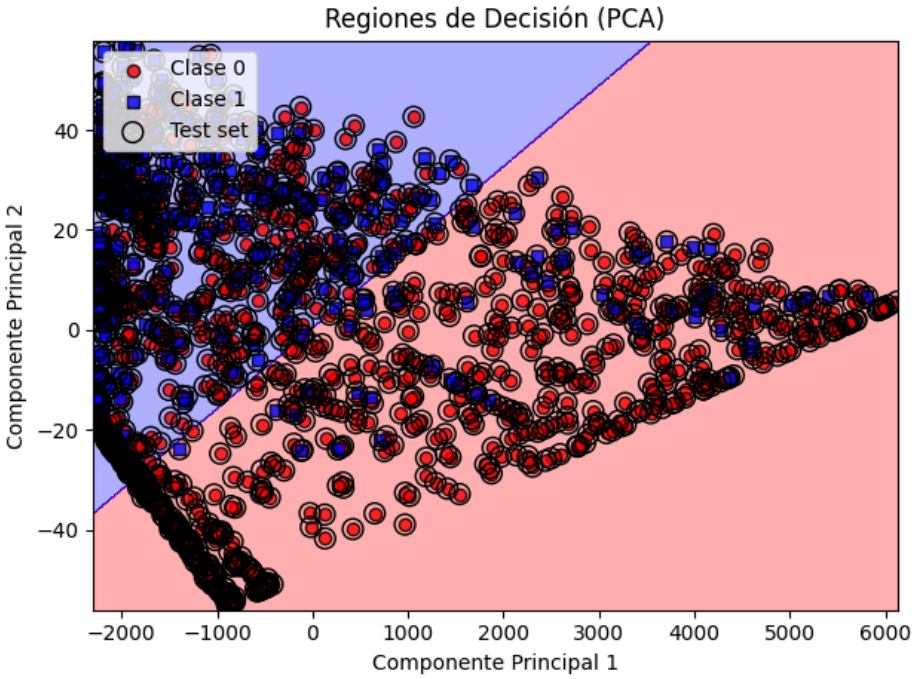
- Sin embargo, dado el bajo recall, esto sugiere que el modelo es más preciso en predecir clientes que se quedan que en detectar churn.



Regiones de Decisión (PCA)

- La reducción de dimensionalidad con PCA ha permitido visualizar cómo el modelo separa las clases.

- Si se observa una gran superposición entre las clases, esto sugiere que el modelo tiene dificultades para encontrar un límite de decisión claro, lo que explicaría el bajo recall.



6. Modelo Adaline

6.1. Arquitectura del Modelo

El modelo implementado se basa en un clasificador Adaline (Adaptive Linear Neuron) utilizando Stochastic Gradient Descent (SGD) como algoritmo de optimización. Este modelo lineal simple pero potente es efectivo para problemas de clasificación binaria como la predicción de deserción de clientes (churn).

Características principales del modelo:

- Base algorítmica: SGDClassifier con función de pérdida logarítmica
- Regularización: Elasticnet (combinación de L1 y L2)
- Manejo de clases desbalanceadas: Ponderación automática por clase
- Enfoque de entrenamiento: Early stopping con validación interna

6.2. Preprocesamiento de datos

El pipeline implementado incluye las siguientes etapas de preprocesamiento:

```
pipeline = Pipeline([
    ('scaler', StandardScaler()),  # Normalización de características
    ('classifier', SGDClassifier(
         loss="log_loss",  # Usar pérdida logarítmica para clasificación binaria
        penalty="elasticnet",  # Combinación de regularización L1 y L2
         class_weight="balanced",  # Ajustar pesos según la frecuencia de clases
        random_state=42,  # Semilla para reproducibilidad
           early_stopping=True,   # Activar parada temprana para evitar sobreajuste
          validation_fraction=0.1  # Usar 10% de datos de entrenamiento para validación
    ))
])
```

- Escalado de características: Normalización mediante StandardScaler para garantizar que todas las variables contribuyan equitativamente al modelo.
- Estratificación: Preservación de la distribución de clases en los conjuntos de entrenamiento y prueba.
- División de datos: 80% entrenamiento, 20% prueba.

6.3. Optimización de Hiperparámetros

Se realizó una búsqueda de hiperparámetros usando GridSearchCV con validación cruzada de 5 folds. Los parámetros evaluados fueron:

Hiperparámetro	Valores Evaluados	Mejor Valor
alpha	[0.0001, 0.001, 0.005, 0.007, 0.01, 0.03, 0.05, 0.1]	0.007
learning_rate	["optimal", "adaptive"]	adaptive

eta0	[0.01, 0.1]	0.1
l1_ratio	[0.15, 0.5, 0.85]	0.5
max_iter	[1000, 2000]	1000

- Criterio de optimización: Área bajo la curva ROC (ROC-AUC)
- Total de combinaciones: 192 configuraciones distintas
- Total de iteraciones: 960 ajustes (5 folds × 192 combinaciones)

6.4. Interpretación de los Hiperparámetros Óptimos

- alpha=0.007: Fuerza de regularización moderada, balanceando el ajuste a los datos y la generalización.
- learning_rate=adaptive: La tasa de aprendizaje se reduce cuando el gradiente no disminuye suficientemente, mejorando la convergencia.
- eta0=0.1: Tasa de aprendizaje inicial relativamente alta, permitiendo una convergencia más rápida.
- l1_ratio=0.5: Balance perfecto entre regularización L1 (lasso) y L2 (ridge), combinando selección de características y control de magnitudes.
- max_iter=1000: Suficientes iteraciones para convergencia sin sobreajuste

6.5. Consideraciones sobre la Convergencia

La combinación de early stopping, tasa de aprendizaje adaptativa y validación interna (10% de los datos de entrenamiento) asegura una convergencia eficiente y robusta, evitando problemas comunes en modelos SGD como la oscilación o la convergencia prematura.

6.6. Limitaciones técnicas del Modelo ADA

- **Linealidad**: El modelo asume relaciones lineales entre características y objetivo.
- **Sensibilidad a outliers**: A pesar del uso de SGD, el modelo puede verse afectado por valores extremos.
- **Complejidad limitada**: No captura interacciones complejas entre variables sin feature engineering explícito.

7. Modelo Regresión Logística

7.1. Arquitectura del Modelo

El tercer modelo implementado para predecir la tasa de abandono de clientes en empresas de telecomunicaciones es el modelo de **Regresión Logística**, es un tipo de modelo estadístico que se utiliza a menudo para la clasificación y el análisis predictivo. La regresión logística estima la probabilidad de que ocurra un evento, como en el reto propuesto que busca clasificar de manera correcta el abandono de los clientes a un servicio (churn) **[1]**

Características importantes sobre este modelo:

- **Modelo de clasificación binaria:** Es un modelo de clasificación binario lo que hace que este modelo se encargue de predecir si una instancia pertenece a la clase positiva o negativa (1 o 0).
- **Función Sigmoide:** Utiliza la función logística también llamada función Sigmoide. Esta función es una curva en forma de S que puede tomar cualquier valor real entre 0 y 1.

  Si esta curva va a infinito la predicción se convertirá en 1, y si la curva pasa el infinito negativo, la predicción se convertirá en 0.

  Si la salida de la función es mayor que 0.5, podemos clasificar el resultado como 1, y si es menor que 0.5 podemos clasificarlo como 0. **[2]**

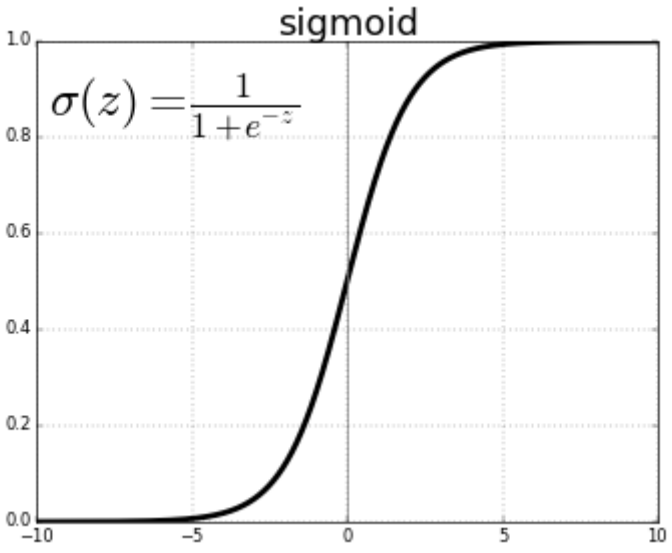


Figura 3.1.1

(Función Sigmoide)

7.2. Proceso de implementación

7.2.1. Importación de librerías

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression

import warnings
warnings.filterwarnings('ignore')

from sklearn.metrics import confusion_matrix, classification_report,
precision_recall_curve, accuracy_score, roc_auc_score
from sklearn.model_selection import GridSearchCV
```

La librería con la que se trabaja es sklearn y también se implementan diferentes funciones que van a ser necesarias para entrenar un modelo de regresión logística.

7.2.2. Definir variable dependiente e independientes

```
X = df.drop('Churn', axis=1)  # Drop the 'Churn' column from
features
y = df['Churn'] # Select the 'Churn' column as the target
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42, stratify=y)
```

Las variables independientes son todas las columnas del dataset Telco-Customer-Churn-V2.csv excepto la columna “Churn” que es nuestra variable de interés, por esta misma razón se define como variable dependiente y por último las variables train y test para entrenar al modelo.

7.2.3. Entrenamiento y evaluación de métricas

```
def metricas(real, predecido):
    print(classification_report(real, predecido))

    cm = confusion_matrix(real, predecido)
    plt.figure(figsize=(8,5))
    sns.heatmap(cm, annot=True, fmt='.2f', xticklabels = ['Servicio
activo', 'Servicio no activo'], yticklabels = ['Servicio activo',
```

```
    'Servicio no activo'])

        plt.ylabel('Real')
        plt.xlabel('Predecido')

        plt.show()

    reg_log = LogisticRegression(random_state=42)
    reg_log.fit(X_train, y_train)

    y_pred_train = reg_log.predict(X_train)
    metricas(y_train, y_pred_train)
```

Se crea una función la cual permite ver la matriz de confusión junto al reporte de clasificación del modelo con los datos de test y train ya implementados. Seguido a esto se entrena el modelo con los datos de entrenamiento que se definieron anteriormente y se prueba este mismo para definir y evaluar las predicciones arrojadas.

7.2.4. Optimización de Hiperparámetros

```
param_grid = {
    'penalty': ['l1', 'l2'],  # Tipo de regularización
    'C': [0.001, 0.01, 0.1, 1, 10, 100],  # Inversa de la fuerza de
regularización
    'solver': ['liblinear', 'saga'],  # Algoritmo de optimización
    'max_iter': [100, 500, 1000] # Número máximo de iteraciones
 }
```

Acá se define un diccionario que especifica el espacio de búsqueda de hiperparametros, variando entre distintos valores, las keys son los hiperparámetros que se quieren ajustar y los valores de la lista son los posibles valores que se prueban para cada uno.

```
grid_search = GridSearchCV(
    estimator=LogisticRegression(random_state=42),  # Tu modelo
    param_grid=param_grid,  # Espacio de búsqueda
    scoring='roc_auc',  # Métrica de evaluación
    cv=5,  # Número de folds para validación cruzada
    n_jobs=-1  # Usar todos los núcleos del procesador
 )
```

Se crea un objeto de GridSearchCV, se le pasa el modelo al que se le van a ajustar los hiperparámetros, el espacio de búsqueda que ya se definió, se define también como métrica de evaluación la curva ROC-AUC, se indica que se utilizará una validación cruzada de 5 folds, lo que

significa que los datos se dividieran en 5 partes y se evaluaran 5 veces y por último se indica que se usen todos los núcleos del procesador para que el trabajo de búsqueda sea más eficaz.

7.3. Interpretación de los Hiperparámetros Óptimos

7.3.1. Glosario

- **C1**: Este es el valor del hiperparámetro 'C' que resultó en el mejor rendimiento. 'C' es el inverso de la fuerza de regularización. Un valor de 1 significa que una regularización moderada fue óptima.
- **Penalty**: Esto indica que el método de regularización L1 fue elegido como la penalización de mejor rendimiento. La regularización L1 puede llevar a modelos dispersos, donde algunos coeficientes de características son cero.
- **Random State**: Esto se estableció en el modelo LogisticRegression inicial pasado a GridSearchCV y se conserva en el mejor estimador. Asegura la reproducibilidad del entrenamiento del modelo.
- **Solver**: Este es el algoritmo utilizado para optimizar el modelo de Regresión Logística. 'liblinear' es una buena opción para conjuntos de datos más pequeños y cuando se usa la regularización L1.

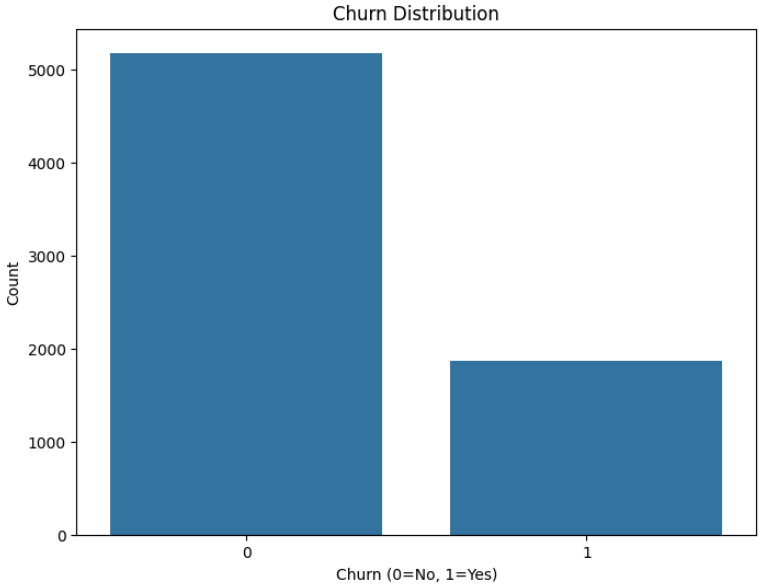
7.3.2. Evaluación Comparativa

Métrica	Perceptron	Adaline	Regresión Logística
Accuracy	**0.77**	**0.74**	**0.74**
Recall	**0.25**	**0.74**	**0.74**
F1-Score	**0.37**	**0.75**	**0.75**
AUC-ROC	**0.80**	**0.83**	**0.73**

7.3.3. Interpretación de Resultados

7.3.3.1. Accuracy

El modelo Perceptrón logró el valor más alto (0.77), indicando que globalmente clasifica correctamente un mayor porcentaje de observaciones. Sin embargo, accuracy puede ser engañosa si hay desequilibrio de clases. (Lo cual hay)



Grafica 3: Distribucion de churn real

7.3.3.2. Recall (Sensibilidad al Churn)

Adaline y Regresión Logística presentaron la sensibilidad más alta (0.74), lo cual es crucial dado que la clase minoritaria (clientes que abandonan la compañía) suele ser de interés prioritario en predicciones de Churn.

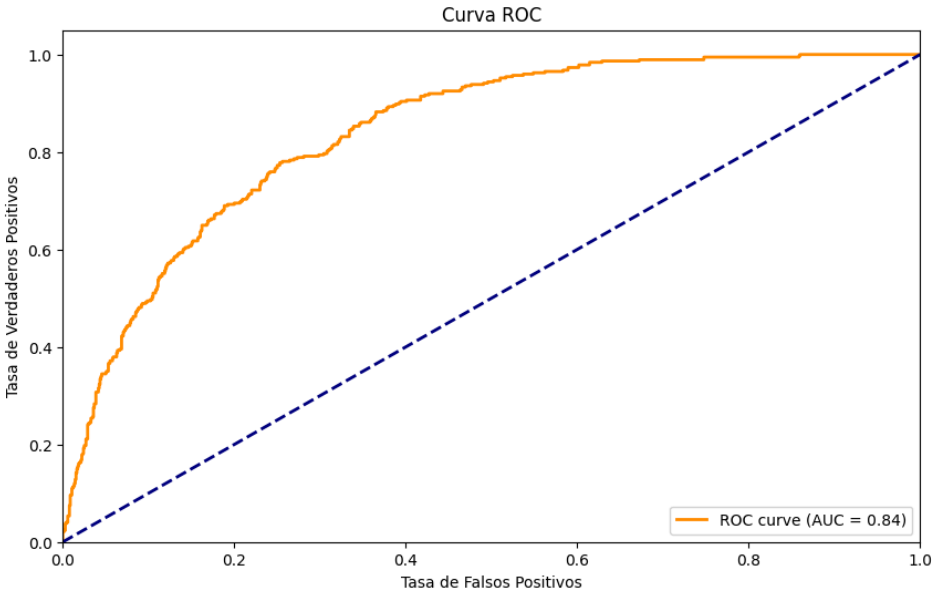
El perceptrón obtuvo un recall bajo (0.25), indicando dificultad para detectar clientes en riesgo de abandonar el servicio.

7.3.3.3. F1-Score

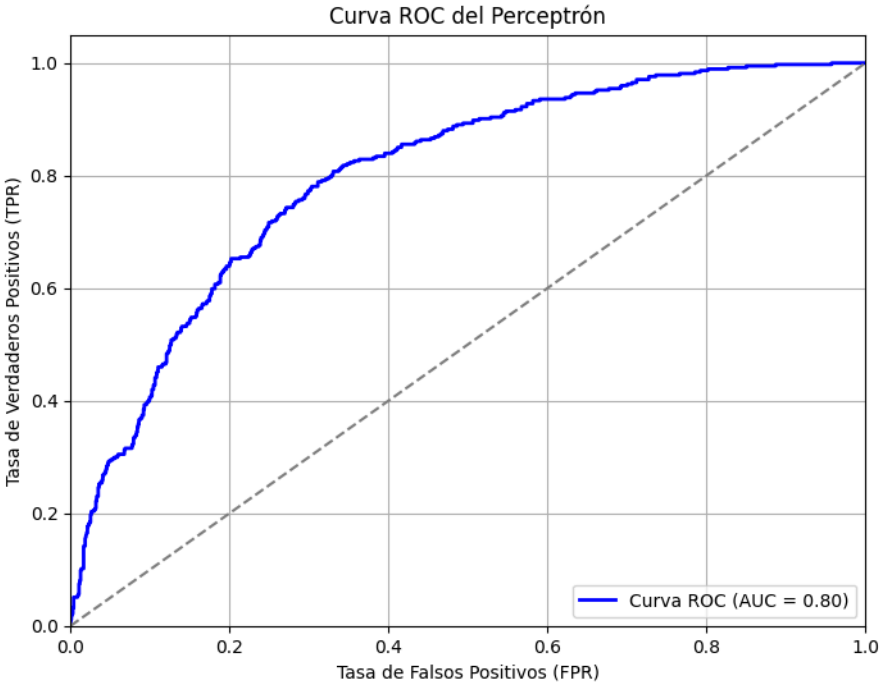
Adaline y Regresión Logística presentaron F1-scores más altos (0.75), reflejando un mejor balance entre precisión y sensibilidad, lo cual las hace más adecuadas para identificar correctamente a clientes propensos al churn.

7.3.3.4. AUC-ROC

Adaline obtuvo el valor más alto (0.83), lo cual indica una mejor capacidad general para distinguir entre clientes que se van y aquellos que se quedan, independientemente del umbral escogido.



Curva ROC Adaline

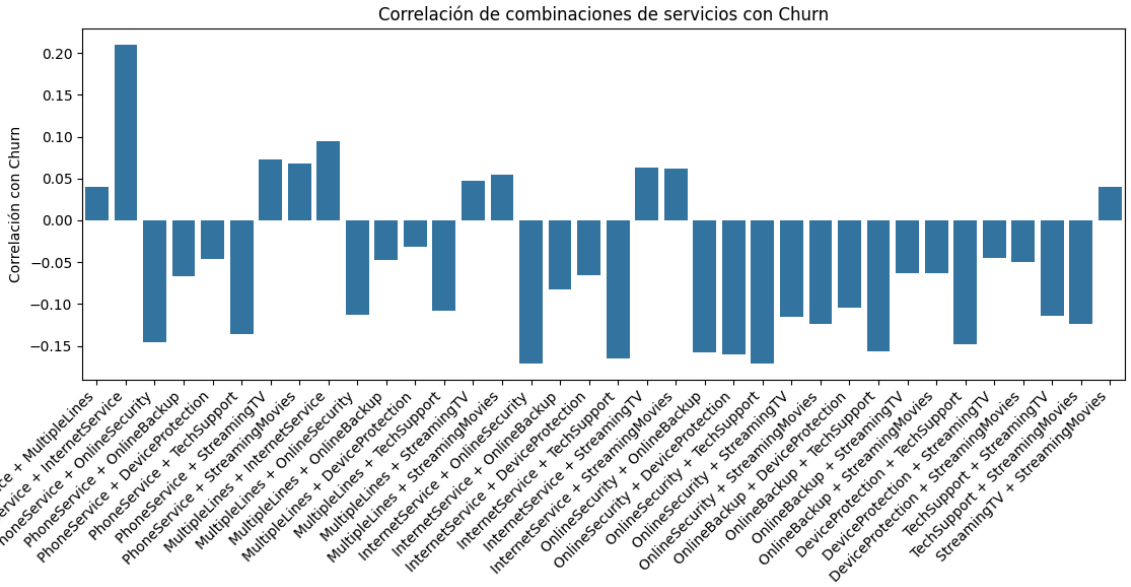


Curva ROC Perceptron

8. Preguntas

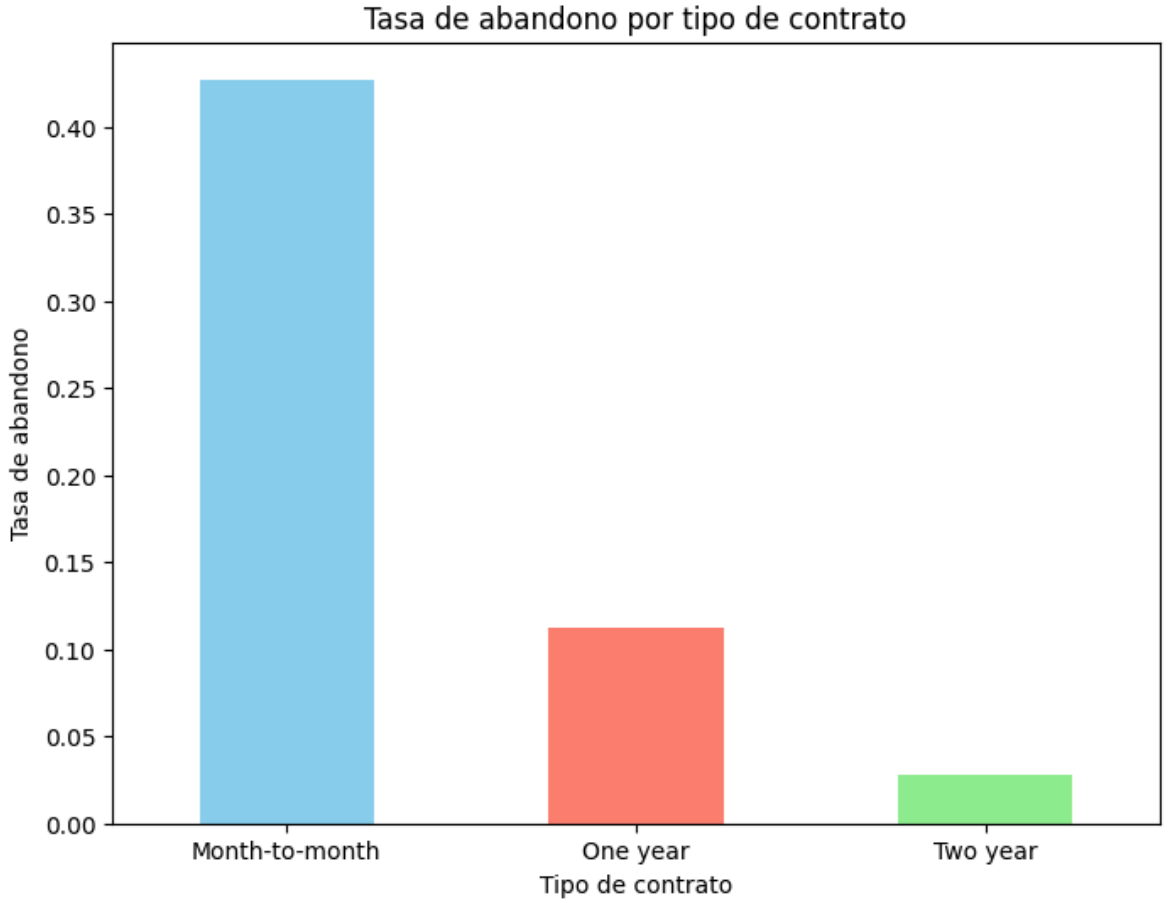
¿Qué combinación de servicios (ej: Fibra Óptica + Streaming) muestra mayor correlación con churn?

La combinación de servicios con mayor correlación a churn es 'Phone Service + Internet Service' con una correlación de 0.21.



¿Cómo varía la probabilidad de abandono entre contratos mensuales vs anuales?

```
Tasa de abandono por tipo de contrato:
Contract
Month-to-month    0.427097
One year          0.112695
Two year          0.028319
Name: Churn, dtype: float64
```



La probabilidad de abandono varía significativamente según el tipo de contrato. Tal como se observa en la imagen, los clientes con contrato mensual presentan una tasa de abandono considerablemente más alta (42.7%) en comparación con aquellos que tienen contratos anuales (11.3%). Esta diferencia sugiere que los contratos mensuales están asociados a una mayor inestabilidad en la retención del cliente, posiblemente debido a la mayor flexibilidad que ofrecen para cancelar el servicio en cualquier momento. En contraste, los contratos anuales parecen fomentar un mayor compromiso por parte del cliente, reduciendo así la probabilidad de abandono.

¿Qué modelo logró mejor recall para la clase minoritaria (Churn=Yes) y por qué?

El modelo Adaline (junto con Regresión Logística) logró el mejor recall (0.74) para la clase minoritaria. Este resultado ocurre porque estos modelos, al optimizar funciones de coste continuas (en Adaline) o probabilísticas (Regresión Logística), logran identificar mejor los patrones asociados al abandono del cliente, aumentando así la sensibilidad para detectar correctamente a clientes en riesgo de churn.

¿Qué 3 features mostraron mayor peso en la regresión logística?

Las 3 *features* que mostraron mayor peso en la regresión logística fueron:

1. Phone Service con un coeficiente de -0.886
2. Contract con un coeficiente de -0.757
3. Online Security con un coeficiente de -0.573

Esto indica que estos tres factores son los más influyentes en la predicción de abandono (Churn) en tu modelo. Como los coeficientes son negativos, sugiere que tener estos servicios activos se asocia con una menor probabilidad de abandono.

Propón un plan de retención basado en tus hallazgos

Plan de Retención Mejorado

Objetivo:

Reducir significativamente la tasa de abandono (churn) de clientes, enfocándose especialmente en aquellos que utilizan combinaciones de servicios identificadas como de alto riesgo, en particular la combinación "Phone Service + Internet Service" que presenta la mayor correlación con churn.

Estrategias:

1. Ofertas y promociones personalizadas:

- Identificar proactivamente a clientes con alto riesgo de abandono (especialmente usuarios que combinan Phone Service e Internet Service) para ofrecerles descuentos exclusivos, promociones atractivas o incentivos en la tarifa mensual.

- Considerar ofrecer pruebas gratuitas o beneficios adicionales en nuevos servicios para aumentar la percepción de valor del cliente.

2. Mejora integral en la experiencia del cliente:

- Simplificar y transparentar el proceso de facturación para evitar confusión y descontento.
- Asegurar un servicio al cliente ágil, amable y resolutivo, reduciendo tiempos de espera y mejorando la comunicación.
- Desarrollar programas efectivos de fidelización que reconozcan y premien la permanencia de los clientes.
- Agregar contenido exclusivo o funcionalidades adicionales a clientes que utilizan múltiples servicios, reforzando su satisfacción y sentido de exclusividad.
- Contactar proactivamente a clientes detectados en riesgo, anticipándose a posibles inconformidades y ofreciéndoles soluciones personalizadas antes de que decidan abandonar.

3. Personalización de servicios:

- Diseñar paquetes personalizados basados en análisis detallados de necesidades, preferencias y patrones de uso de cada segmento de clientes.
- Adaptar continuamente estas ofertas basándose en feedback directo de los clientes para mantenerlas relevantes y atractivas.

4. Seguimiento constante de la satisfacción del cliente:

- Implementar encuestas periódicas de satisfacción para detectar tempranamente áreas críticas que requieren mejoras.
- Analizar de manera regular los comentarios y sugerencias de los clientes para entender mejor las causas detrás de la insatisfacción y abordarlas rápidamente.
- Segmentar a los clientes según sus características específicas para optimizar la efectividad de las estrategias de retención.

Seguimiento y Evaluación:

- Establecer monitoreos periódicos de la tasa de abandono (churn) para medir el impacto de las acciones implementadas.

- Evaluar regularmente los resultados obtenidos y ajustar dinámicamente las estrategias de retención basándose en los datos y la retroalimentación de los clientes, garantizando una mejora continua en la experiencia del usuario y en la eficacia del plan de retención.

9. Referencias

IBM. (n.d.). ¿Qué es la regresión logística? *Ibm.com*. Retrieved March 31, 2025, from https://www.ibm.com/mx-es/topics/logistic-regression

Gonzalez, L. (2019, June 28). *Regresión Logística - Teoría*. Aprende IA. https://aprendeia.com/algoritmo-regresion-logistica-machine-learning-teoria/

Rosebrock, A. (2021, May 6). *Implementing the Perceptron Neural Network with Python*. PyImageSearch. https://pyimagesearch.com/2021/05/06/implementing-the-perceptron-neural-network-with-python/

Natsu6767. (2018). *GitHub - Natsu6767/Adaline: Implementation of an Adaptive Linear Neuron in python.* GitHub. https://github.com/Natsu6767/Adaline