📊 Guía Completa: Análisis y Predicción de Customer Churn en Telecomunicaciones

---

📖 Información del Documento

Proyecto: Predicción de Abandono de Clientes (Customer Churn)
Industria: Telecomunicaciones
Dataset: 7,043 clientes con 21 variables
Objetivo: Predecir qué clientes tienen alta probabilidad de abandonar el servicio
Metodología: Machine Learning con enfoque en clasificación binaria

---

📑 Tabla de Contenidos

🎯 Resumen Ejecutivo

Este documento presenta un análisis completo de predicción de churn (abandono de clientes) para una empresa de telecomunicaciones. A través de un proceso estructurado de ciencia de datos, se desarrolló un modelo de Machine Learning capaz de:

- ✅ **Detectar el 82% de los clientes en riesgo** de abandonar el servicio
- ✅ **Generar un ROI positivo** de aproximadamente $205,000
- ✅ **Identificar factores clave** que influyen en el churn
- ✅ **Proporcionar recomendaciones accionables** para estrategias de retención

Hallazgos Clave:

- 27% de los clientes abandonan el servicio
- Los contratos mes a mes tienen ~42% de churn vs. 3% en contratos de 2 años
- Clientes nuevos (< 12 meses) tienen el mayor riesgo
- Precios altos y falta de servicios adicionales aumentan el churn

---

Bloque 1: Introducción y Descripción del Proyecto

📋 Descripción General

Este primer bloque del notebook es como la **portada y el índice de un libro**: nos presenta el proyecto completo, establece las expectativas y nos da un mapa del viaje que vamos a emprender en el análisis de datos.

🎯 Propósito y Objetivo

El objetivo principal de este bloque es:

1. **Presentar el problema de negocio**: Predicción del abandono de clientes (Customer Churn) en una empresa de telecomunicaciones
2. **Establecer la metodología**: Definir los pasos que seguiremos en el análisis
3. **Describir los datos**: Dar una visión general del dataset que vamos a utilizar

¿Por qué es importante?

Imagina que vas a construir una casa. Antes de empezar, necesitas:

- Un plano (metodología)
- Saber qué materiales tienes (dataset)
- Entender qué tipo de casa quieres construir (objetivo)

Este bloque es exactamente eso: el plano maestro de nuestro proyecto.

🔑 Conceptos Clave

1. **Customer Churn (Abandono de Clientes)**

¿Qué es?

El "churn" es cuando un cliente decide dejar de usar los servicios de una empresa. Es como cuando cancelas tu suscripción de Netflix o cambias de compañía telefónica.

¿Por qué importa?

- Conseguir un cliente nuevo cuesta entre 5 y 25 veces más que retener uno existente
- Predecir quién se va a ir permite tomar acciones preventivas (descuentos, mejores ofertas, atención personalizada)

Analogía: Es como un médico que puede predecir una enfermedad antes de que aparezca, permitiendo tratamiento preventivo.

2. Machine Learning para Predicción

El proyecto utiliza algoritmos de aprendizaje automático que “aprenden” de datos históricos para predecir comportamientos futuros.

Analogía: Es como enseñarle a un niño a reconocer frutas mostrándole muchas manzanas, naranjas y plátanos. Después de ver suficientes ejemplos, puede identificar una fruta nueva que nunca ha visto.

3. Metodología del Proyecto

El notebook sigue un proceso estructurado de 7 pasos:

Paso 1: Análisis Exploratorio de Datos (EDA)

- **¿Qué hace?** Explora y entiende los datos
- **Analogía**: Como un detective que examina todas las pistas antes de resolver un caso

Paso 2: Preprocesamiento

- **¿Qué hace?** Limpia y prepara los datos
- **Analogía**: Como lavar y cortar verduras antes de cocinar

Paso 3: Feature Engineering

- **¿Qué hace?** Crea nuevas variables útiles a partir de las existentes
- **Analogía**: Como un chef que combina ingredientes básicos para crear nuevos sabores

Paso 4: Modelado

- **¿Qué hace?** Entrena diferentes algoritmos de predicción
- **Analogía**: Como probar diferentes recetas para ver cuál sabe mejor

Paso 5: Optimización

- **¿Qué hace?** Ajusta los modelos para mejorar su rendimiento
- **Analogía**: Como afinar un instrumento musical para que suene perfecto

Paso 6: Evaluación

- **¿Qué hace?** Mide qué tan bien funcionan los modelos
- **Analogía**: Como calificar un examen para ver qué tan bien aprendiste

Paso 7: Interpretabilidad

- **¿Qué hace?** Entiende qué factores son más importantes para la predicción
- **Analogía**: Como descubrir qué ingrediente hace que una receta sea especial

---

📊 Descripción del Dataset

El dataset contiene información de **7,043 clientes** con **21 variables**:

Tipos de Información

1. **Demográfica**: Quiénes son los clientes
    - Género (masculino/femenino)
    - Edad (si son adultos mayores)
    - Si tienen pareja o dependientes
2. **Servicios Contratados**: Qué usan
    - Servicio telefónico
    - Internet (DSL o Fibra óptica)

    - Servicios adicionales (streaming, seguridad online, etc.)
3. **Información de Cuenta**: Cómo pagan
    - Tipo de contrato (mensual, anual, bianual)
    - Método de pago
    - Cargos mensuales y totales
4. **Variable Objetivo**: Lo que queremos predecir
    - **Churn**: Si el cliente se fue (Yes) o se quedó (No)

🔗 Relación con el Análisis General

Este bloque es el **punto de partida** del proyecto. Establece:

- **El problema**: ¿Qué queremos resolver?
- **El camino**: ¿Cómo lo vamos a resolver?
- **Los recursos**: ¿Con qué datos contamos?

Sin esta introducción, estaríamos navegando sin brújula. Cada bloque posterior del notebook se construye sobre esta base.

💡 Puntos Clave para Recordar

1. **Customer Churn** es un problema crítico de negocio que cuesta mucho dinero a las empresas
2. El proyecto sigue una **metodología estructurada** de 7 pasos
3. Tenemos **7,043 clientes** con **21 variables** para analizar
4. El objetivo final es **predecir** qué clientes tienen alta probabilidad de irse
5. Esta predicción permite tomar **acciones preventivas** para retener clientes

---

🎓 Conclusión

Este bloque introductorio es como el mapa de un tesoro: nos muestra dónde estamos, a dónde vamos y qué camino seguiremos. Establece las bases para todo el análisis posterior y nos ayuda a entender el valor de negocio del proyecto.

Siguiente paso: Importar las herramientas (librerías) que necesitaremos para realizar el análisis.

Bloque 2: Importación de Librerías

📋 Descripción General

Este bloque es como **preparar la caja de herramientas** antes de comenzar un trabajo. Importa todas las librerías (bibliotecas de código) necesarias para realizar el análisis de datos, crear visualizaciones, entrenar modelos y evaluar resultados.

🎯 Propósito y Objetivo

El objetivo de este bloque es:

1. **Importar todas las herramientas necesarias** para el proyecto
2. **Configurar el entorno de trabajo** (suprimir advertencias, configurar visualizaciones)
3. **Verificar que todo está listo** para comenzar el análisis

¿Por qué es importante?

Analogía del carpintero: Imagina que eres un carpintero que va a construir una mesa. Antes de empezar, necesitas sacar del taller:

- El martillo (para clavar)
- La sierra (para cortar)
- El nivel (para medir)
- El taladro (para perforar)

De la misma manera, este bloque "saca del taller" todas las herramientas de software que necesitaremos.

🔑 Conceptos Clave y Librerías Importadas

1. **Librerías de Manipulación de Datos**

NumPy *(import numpy as np)*

- **¿Qué hace?** Maneja operaciones matemáticas y arrays numéricos
- **Analogía**: Es como una calculadora científica súper potente
- **Uso en el proyecto**: Cálculos matemáticos, manejo de valores faltantes (NaN)

Pandas *(import pandas as pd)*

- **¿Qué hace?** Manipula y analiza datos en formato de tablas (DataFrames)
- **Analogía**: Es como Excel pero con superpoderes
- **Uso en el proyecto**: Cargar el CSV, limpiar datos, crear nuevas columnas

---

2. **Librerías de Visualización**

Matplotlib *(import matplotlib.pyplot as plt)*

- **¿Qué hace?** Crea gráficos básicos (líneas, barras, dispersión)
- **Analogía**: Es como un lienzo y pinceles para pintar gráficos
- **Uso en el proyecto**: Crear visualizaciones personalizadas

Seaborn *(import seaborn as sns)*

- **¿Qué hace?** Crea gráficos estadísticos más elegantes y complejos
- **Analogía**: Es como Matplotlib pero con plantillas profesionales pre-diseñadas
- **Uso en el proyecto**: Gráficos de distribución, correlaciones, comparaciones

---

3. **Librerías de Preprocesamiento**

train_test_split

- **¿Qué hace?** Divide los datos en conjuntos de entrenamiento y prueba
- **Analogía**: Como dividir un mazo de cartas en dos grupos: uno para practicar y otro para el examen final

StandardScaler

- **¿Qué hace?** Normaliza los datos para que estén en la misma escala
- **Analogía**: Como convertir todas las medidas a la misma unidad (metros en vez de mezclar metros, centímetros y kilómetros)

LabelEncoder

- **¿Qué hace?** Convierte categorías de texto en números
- **Analogía**: Como asignar números a colores (Rojo=1, Azul=2, Verde=3)

ColumnTransformer y Pipeline

- **¿Qué hace?** Crea flujos de trabajo automatizados para procesar datos
- **Analogía**: Como una línea de ensamblaje en una fábrica donde cada estación hace una tarea específica

---

4. **Librerías de Modelos de Machine Learning**

Logistic Regression *(Regresión Logística)*

- **¿Qué hace?** Modelo simple para clasificación binaria (Sí/No)
- **Analogía**: Como trazar una línea para separar dos grupos

Decision Tree *(Árbol de Decisión)*

- **¿Qué hace?** Toma decisiones siguiendo una serie de preguntas
- **Analogía**: Como un diagrama de flujo de “si esto, entonces aquello”

Random Forest *(Bosque Aleatorio)*

- **¿Qué hace?** Combina muchos árboles de decisión
- **Analogía**: Como pedir opinión a 100 expertos y tomar la decisión por mayoría

Gradient Boosting

- **¿Qué hace?** Construye modelos secuencialmente, cada uno corrigiendo errores del anterior
- **Analogía**: Como un estudiante que aprende de sus errores en cada examen de práctica

SVC *(Support Vector Classifier)*

- **¿Qué hace?** Encuentra el mejor límite para separar clases
- **Analogía**: Como encontrar la mejor valla para separar dos rebaños de ovejas

KNeighbors *(K-Vecinos Más Cercanos)*

- **¿Qué hace?** Clasifica basándose en los vecinos más cercanos
- **Analogía**: “Dime con quién andas y te diré quién eres”

XGBoost

- **¿Qué hace?** Versión optimizada y potente de Gradient Boosting
- **Analogía**: Como Gradient Boosting pero con turbo

---

5. **Librerías de Métricas de Evaluación**

Estas herramientas miden qué tan bien funcionan nuestros modelos:

- **accuracy_score**: ¿Cuántas predicciones fueron correctas?
- **precision_score**: De las predicciones positivas, ¿cuántas fueron correctas?
- **recall_score**: De todos los casos positivos reales, ¿cuántos detectamos?
- **f1_score**: Balance entre precisión y recall
- **confusion_matrix**: Tabla que muestra aciertos y errores
- **roc_auc_score**: Mide la capacidad de discriminación del modelo

Analogía del detector de metales:

- **Accuracy**: ¿Cuántas veces acertó en total?
- **Precision**: Cuando pita, ¿realmente hay metal?
- **Recall**: De todos los metales enterrados, ¿cuántos encontró?

---

6. **Librerías para Manejo de Desbalanceo**

***SMOTE** (Synthetic Minority Over-sampling Technique)*

- **¿Qué hace?** Crea ejemplos sintéticos de la clase minoritaria
- **Analogía**: Si tienes 100 fotos de gatos y solo 10 de perros, SMOTE crea más fotos de perros (sintéticas) para balancear

RandomOverSampler

- **¿Qué hace?** Duplica aleatoriamente ejemplos de la clase minoritaria
- **Analogía**: Fotocopiar las 10 fotos de perros varias veces

RandomUnderSampler

- **¿Qué hace?** Reduce ejemplos de la clase mayoritaria
- **Analogía**: Eliminar algunas de las 100 fotos de gatos para igualar

---

7. **Librerías de Optimización**

GridSearchCV

- **¿Qué hace?** Prueba todas las combinaciones posibles de parámetros
- **Analogía**: Como probar todas las combinaciones de una cerradura hasta encontrar la correcta

RandomizedSearchCV

- **¿Qué hace?** Prueba combinaciones aleatorias de parámetros (más rápido)
- **Analogía**: En vez de probar todas las combinaciones, prueba algunas al azar

---

🎨 Configuración de Visualización

El bloque también configura cómo se verán los gráficos:

```
plt.style.use('seaborn-v0_8-darkgrid')  # Estilo visual
sns.set_palette("husl")                 # Paleta de colores
plt.rcParams['figure.figsize'] = (12, 6) # Tamaño de gráficos
plt.rcParams['font.size'] = 10          # Tamaño de letra
```

Analogía: Como configurar el tema de tu teléfono (modo oscuro, colores, tamaño de letra).

🔗 Relación con el Análisis General

Este bloque es **fundamental** porque:

1. **Sin estas herramientas, no podríamos hacer nada** - Es como intentar cocinar sin utensilios
2. **Establece el entorno de trabajo** - Todo lo que viene después depende de estas importaciones
3. **Organiza las herramientas por categoría** - Facilita encontrar lo que necesitamos

---

💡 Puntos Clave para Recordar

1. Las **librerías son herramientas** que nos ahorran escribir código desde cero
2. Cada librería tiene un **propósito específico** (datos, visualización, modelos, métricas)
3. La **configuración inicial** asegura que todo funcione correctamente
4. Este bloque es **preparación**, no análisis - Es como afilar los cuchillos antes de cocinar

---

🎓 Conclusión

Este bloque prepara todo el arsenal de herramientas que necesitaremos. Es breve pero crítico: sin él, ningún análisis posterior sería posible. Es la base técnica sobre la que se construye todo el proyecto.

Siguiente paso: Cargar los datos y hacer una exploración inicial.

Bloque 3: Carga y Exploración Inicial de Datos

📋 Descripción General

Este bloque es el **primer contacto real con los datos**. Es como abrir una caja misteriosa para ver qué hay dentro. Aquí cargamos el archivo CSV con la información de los clientes y hacemos una inspección inicial para entender su estructura, tamaño y contenido.

🎯 Propósito y Objetivo

Los objetivos principales de este bloque son:

1. **Cargar el dataset** desde el archivo CSV de manera robusta
2. **Verificar las dimensiones** (cuántas filas y columnas tiene)
3. **Inspeccionar las primeras filas** para ver cómo lucen los datos
4. **Identificar los tipos de datos** de cada columna
5. **Obtener estadísticas descriptivas** básicas

¿Por qué es importante?

Analogía del médico: Antes de diagnosticar a un paciente, el médico necesita:

- Conocer sus datos básicos (edad, peso, altura)
- Ver su historial médico
- Hacer un examen físico inicial

De la misma manera, antes de analizar los datos, necesitamos conocerlos, verlos y entender su estructura básica.

🔑 Conceptos Clave y Técnicas Utilizadas

1. **Función `cargar_datos()` - Carga Robusta**

El código crea una función personalizada que intenta cargar el archivo desde múltiples ubicaciones posibles.

¿Por qué hacer esto?

Analogía: Es como buscar tus llaves en varios lugares donde podrían estar (bolsillo, mesa, bolso) en vez de asumir que están en un solo lugar.

Beneficios:

- Funciona en diferentes entornos (Google Colab, local, servidor)
- Evita errores por rutas incorrectas
- Hace el código más portable y robusto

2. **Carga del CSV con Pandas**

pd.read_csv() lee el archivo CSV y lo convierte en un DataFrame de Pandas.

Analogía: Es como escanear un documento físico y convertirlo en un archivo digital que puedes editar.

3. **Inspección de Dimensiones**

Resultado: (7043, 21)

- **7,043 filas** = 7,043 clientes
- **21 columnas** = 21 variables por cliente

Analogía: Es como saber que tienes un álbum de fotos con 7,043 páginas y cada página tiene 21 datos diferentes.

4. **Tipos de Datos (`dtypes`)**

Tipos principales encontrados:

- **object**: Texto (categorías como “Yes”, “No”, “Male”, “Female”)
- **int64**: Números enteros (como tenure: 1, 2, 34, 45)
- **float64**: Números decimales (como MonthlyCharges: 29.85, 56.95)

Observación importante: TotalCharges aparece como **object** (texto) cuando debería ser numérico. ¡Esto es una señal de alerta!

5. Estadísticas Descriptivas (`df.describe()`)

Para variables numéricas, calcula:

- **count**: Cantidad de valores
- **mean**: Promedio
- **std**: Desviación estándar
- **min/max**: Valores mínimo y máximo
- **25%, 50%, 75%**: Cuartiles

Ejemplo con `tenure` (meses como cliente):

- **Promedio**: 32.37 meses (~2.7 años)
- **Mínimo**: 0 meses (clientes nuevos)
- **Máximo**: 72 meses (6 años)

📊 Hallazgos Clave de la Exploración Inicial

Dimensiones del Dataset

- ✅ 7,043 clientes
- ✅ 21 variables
- ✅ Tamaño manejable para análisis

Tipos de Variables

1. **Variables Demográficas**:
   - `gender`: Género (Male/Female)
   - `SeniorCitizen`: Si es adulto mayor (0/1)
   - `Partner`: Tiene pareja (Yes/No)
   - `Dependents`: Tiene dependientes (Yes/No)
2. **Variables de Servicio**:
   - `PhoneService`: Servicio telefónico
   - `InternetService`: Tipo de internet (DSL/Fiber optic/No)
   - Servicios adicionales: OnlineSecurity, OnlineBackup, etc.
3. **Variables de Cuenta**:
   - `tenure`: Meses como cliente
   - `Contract`: Tipo de contrato
   - `PaymentMethod`: Método de pago
   - `MonthlyCharges`: Cargo mensual
   - `TotalCharges`: Cargo total
4. **Variable Objetivo**:
   - `Churn`: Si el cliente se fue (Yes/No)

Problema Detectado

- ⚠️ `TotalCharges` está como texto (object) en vez de número
- Esto indica que hay valores no numéricos que necesitaremos investigar y limpiar

---

🔗 Relación con el Análisis General

Este bloque es el **punto de partida del análisis de datos**:

1. **Confirma que tenemos los datos** correctamente cargados
2. **Identifica la estructura** que trabajaremos
3. **Detecta problemas iniciales** (como TotalCharges)
4. **Establece el contexto** para la limpieza y análisis posterior

---

💡 Puntos Clave para Recordar

1. **Carga robusta**: El código busca el archivo en múltiples ubicaciones
2. **7,043 clientes** con **21 variables** cada uno
3. **Tres tipos de datos**: object (texto), int64 (enteros), float64 (decimales)
4. **Problema detectado**: TotalCharges debería ser numérico pero está como texto
5. **Estadísticas iniciales**: Los clientes tienen en promedio 32 meses de antigüedad

---

🎓 Conclusión

Este bloque es como el **reconocimiento del terreno** antes de construir. Nos da una visión panorámica de los datos: qué tenemos, cómo está estructurado y qué problemas potenciales existen.

Siguiente paso: Analizar la calidad de los datos en profundidad y detectar valores faltantes o inconsistencias.

Bloque 4: Análisis de Calidad de Datos

📋 Descripción General

Este bloque es como una **inspección de calidad en una fábrica**. Después de cargar los datos, necesitamos verificar que estén en buen estado: buscar valores faltantes, detectar inconsistencias y corregir problemas antes de continuar con el análisis.

🎯 Propósito y Objetivo

Los objetivos principales de este bloque son:

1. **Detectar valores faltantes** (missing values) en el dataset
2. **Identificar anomalías** en los tipos de datos
3. **Investigar el problema de TotalCharges** detectado anteriormente
4. **Limpiar y corregir** los datos problemáticos
5. **Verificar** que los datos estén listos para el análisis

¿Por qué es importante?

Analogía de la cocina: Imagina que vas a preparar una ensalada. Antes de cocinar, necesitas:

- Revisar que todas las verduras estén frescas (no falten ingredientes)
- Lavar y limpiar lo que esté sucio
- Desechar lo que esté en mal estado

Los datos son igual: necesitan limpieza antes de usarlos.

🔑 Conceptos Clave y Técnicas Utilizadas

1. **Detección de Valores Faltantes**

```
df.isnull().sum()  # Cuenta valores nulos por columna
```

¿Qué son valores faltantes?

- Datos que no existen o no fueron registrados
- En Pandas se representan como `NaN` (Not a Number) o `None`

Analogía: Es como tener un formulario donde algunas personas dejaron preguntas en blanco.

Resultado inicial: ¡No hay valores `NaN` explícitos! Pero...

---

2. **Investigación del Problema de TotalCharges**

El bloque anterior detectó que `TotalCharges` está como texto (object) en vez de número. Este bloque investiga por qué:

```
df['TotalCharges'].dtype  # Retorna: object
```

Descubrimiento clave: Hay **11 registros** con espacios en blanco (' ') en lugar de números.

Analogía: Es como encontrar que en 11 formularios, en vez de escribir un número en "Total a pagar", dejaron un espacio vacío.

---

3. **Análisis de Registros Problemáticos**

El código examina estos 11 registros:

```
espacios_blancos = df[df['TotalCharges'] == ' ']
```

Hallazgo importante:

- Todos tienen `tenure = 0` (son clientes nuevos, con 0 meses de antigüedad)
- Tienen `MonthlyCharges` pero no `TotalCharges`

Lógica de negocio: Si un cliente es nuevo (tenure=0), su cargo total debería ser igual a su cargo mensual (aún no ha pagado más de un mes).

Analogía: Si acabas de contratar Netflix hoy, tu pago total hasta ahora es igual a la mensualidad, no más.

---

4. Estrategia de Limpieza

El bloque implementa una solución en 3 pasos:

Paso 1: Convertir espacios en blanco a NaN

```
df['TotalCharges'] = df['TotalCharges'].replace(' ', np.nan)
```

¿Por qué? Porque Pandas maneja mejor los valores `NaN` que los espacios en blanco.

Paso 2: Convertir a numérico

```
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
```

¿Qué hace `pd.to_numeric()`?

- Convierte texto a números
- `errors='coerce'` significa: "si no puedes convertir, pon NaN"

Analogía: Es como un traductor que convierte palabras a números, y si no puede, deja un espacio en blanco.

Paso 3: Imputar valores faltantes

```
df.loc[df['TotalCharges'].isna(), 'TotalCharges'] = \
   df.loc[df['TotalCharges'].isna(), 'MonthlyCharges']
```

¿Qué significa "imputar"?

- Rellenar valores faltantes con valores razonables
- En este caso: TotalCharges = MonthlyCharges para clientes nuevos

Analogía: Es como completar las respuestas en blanco de un formulario usando lógica (si alguien nació en 2000 y estamos en 2025, tiene ~25 años).

---

5. Verificación Final

```
df['TotalCharges'].isna().sum()  # Retorna: 0
df.isnull().sum().sum()          # Retorna: 0
```

Confirmación: ✅ Ya no hay valores faltantes en todo el dataset.

📊 Hallazgos Clave del Análisis de Calidad

Problemas Detectados

1. ⚠️ 11 registros con `TotalCharges` vacío (espacios en blanco)
2. ⚠️ Todos corresponden a clientes nuevos (tenure = 0)
3. ⚠️ `TotalCharges` estaba almacenado como texto en vez de número

Soluciones Aplicadas

1. ✅ Convertir espacios en blanco a NaN
2. ✅ Convertir `TotalCharges` de texto a número (float64)
3. ✅ Imputar valores faltantes usando `MonthlyCharges`
4. ✅ Verificar que no queden valores faltantes

Estado Final

- ✅ 0 valores faltantes en todo el dataset
- ✅ Todos los tipos de datos son correctos
- ✅ Los datos están limpios y listos para análisis

🔗 Relación con el Análisis General

Este bloque es **crítico** porque:

1. **Datos sucios** = **Resultados incorrectos**: Si no limpiamos los datos, los modelos aprenderán patrones incorrectos
2. **Previene errores futuros**: Muchas funciones de análisis fallan con valores faltantes
3. **Mejora la calidad del modelo**: Datos limpios = mejores predicciones

Analogía del edificio: No puedes construir un edificio sólido sobre cimientos débiles. Los datos limpios son los cimientos del análisis.

💡 Puntos Clave para Recordar

1. **Valores faltantes** pueden estar ocultos (como espacios en blanco)
2. **Siempre investigar** por qué faltan datos antes de eliminarlos
3. **Imputación inteligente**: Usar lógica de negocio para rellenar valores
4. **Verificación**: Siempre confirmar que la limpieza funcionó
5. **11 registros** fueron corregidos (0.16% del dataset)
6. **TotalCharges** ahora es numérico y completo

🎓 Conclusión

Este bloque demuestra que la **calidad de datos es fundamental**. Encontramos un problema sutil (espacios en blanco en vez de NaN), lo investigamos, entendimos su causa (clientes nuevos) y aplicamos una solución lógica (igualar a MonthlyCharges).

Lección importante: Los datos del mundo real casi nunca están perfectos. La limpieza de datos es una parte esencial (y a menudo la más larga) de cualquier proyecto de ciencia de datos.

Siguiente paso: Con los datos limpios, podemos proceder al Análisis Exploratorio de Datos (EDA) para entender patrones y relaciones.

Bloque 5: Análisis Exploratorio de Datos (EDA)

📋 Descripción General

Este bloque es como **ser un detective que investiga un caso**. Ahora que los datos están limpios, exploramos en profundidad para descubrir patrones, tendencias y relaciones que nos ayuden a entender por qué los clientes abandonan el servicio.

🎯 Propósito y Objetivo

Los objetivos principales de este bloque son:

1. **Analizar la variable objetivo (Churn)**: ¿Cuántos clientes se van vs. se quedan?
2. **Explorar variables categóricas**: ¿Qué características tienen los clientes que se van?
3. **Analizar variables numéricas**: ¿Hay diferencias en cargos o antigüedad?
4. **Estudiar correlaciones**: ¿Qué variables están relacionadas entre sí?
5. **Generar visualizaciones** que cuenten la historia de los datos

¿Por qué es importante?

Analogía del médico: Antes de recetar un tratamiento, el médico necesita:

- Entender los síntomas
- Identificar patrones
- Buscar causas subyacentes

El EDA es el "diagnóstico" que nos permite entender el problema antes de construir modelos.

🔑 Conceptos Clave y Técnicas Utilizadas

1. **Análisis de la Variable Objetivo: Churn**

Distribución de Churn:

- **No** (se quedaron): ~73% de los clientes
- **Yes** (se fueron): ~27% de los clientes

¿Qué significa esto?

Analogía del restaurante: De cada 100 clientes que entran, 27 no vuelven nunca. Eso es un problema serio que cuesta dinero.

Implicación importante: Hay **desbalanceo de clases**

- Más clientes se quedan que se van
- Esto puede afectar el entrenamiento de modelos (los modelos tienden a predecir la clase mayoritaria)

---

2. **Análisis de Variables Categóricas**

El bloque examina cómo diferentes características se relacionan con el churn:

Género (Gender)

- **Hallazgo**: El churn es similar entre hombres y mujeres
- **Conclusión**: El género NO es un factor determinante

Adultos Mayores (SeniorCitizen)

- **Hallazgo**: Los adultos mayores tienen MAYOR tasa de churn
- **Analogía**: Como si los clientes mayores fueran más propensos a cambiar de proveedor

Tipo de Contrato (Contract)

- **Hallazgo clave**:
    - Contratos mes a mes: ALTA tasa de churn (~42%)
    - Contratos de 1 año: Churn moderado (~11%)
    - Contratos de 2 años: BAJA tasa de churn (~3%)

Analogía del gimnasio: Las personas con membresía mensual cancelan más fácilmente que las que pagaron por todo el año.

Insight de negocio: ¡Ofrecer contratos largos reduce significativamente el churn!

Servicio de Internet (InternetService)

- **Hallazgo**: Clientes con Fibra Óptica tienen MAYOR churn que DSL
- **Posible razón**: Fibra óptica es más cara, los clientes son más sensibles al precio

Servicios Adicionales

- **OnlineSecurity, TechSupport, OnlineBackup**: Los clientes SIN estos servicios tienen mayor churn
- **Analogía**: Es como tener un seguro completo vs. básico; el completo te hace sentir más protegido y menos propenso a cambiar

---

3. **Análisis de Variables Numéricas**

Tenure (Antigüedad en meses)

- **Clientes que se van**: Promedio de ~18 meses
- **Clientes que se quedan**: Promedio de ~38 meses

Hallazgo crítico: Los clientes nuevos son MÁS propensos a irse.

Analogía: Es como una relación: los primeros meses son críticos. Si sobrevives el primer año, es más probable que dures mucho tiempo.

MonthlyCharges (Cargos Mensuales)

- **Clientes que se van**: Pagan MÁS en promedio (~$75)
- **Clientes que se quedan**: Pagan MENOS en promedio (~$61)

Insight: El precio alto es un factor de riesgo para el churn.

TotalCharges (Cargos Totales)

- **Clientes que se van**: Han pagado MENOS en total
- **Razón**: Tienen menos antigüedad (tenure bajo)

---

4. **Análisis de Correlaciones**

El bloque crea una **matriz de correlación** que muestra cómo las variables se relacionan entre sí.

¿Qué es correlación?

- Mide si dos variables se mueven juntas
- Valores de -1 a +1:
    - **+1**: Correlación positiva perfecta (si una sube, la otra también)
    - **0**: No hay relación

- **-1**: Correlación negativa perfecta (si una sube, la otra baja)

Correlaciones importantes encontradas:

1. **TotalCharges ↔️ Tenure**: +0.83 (fuerte positiva)
   - **Lógica**: Más tiempo como cliente = más has pagado en total
2. **MonthlyCharges ↔️ TotalCharges**: +0.65 (moderada positiva)
   - **Lógica**: Si pagas más al mes, pagas más en total
3. **Churn ↔️ Tenure**: Negativa
   - **Lógica**: Más antigüedad = menos probabilidad de irse
4. **Churn ↔️ MonthlyCharges**: Positiva
   - **Lógica**: Más caro = más probabilidad de irse

📊 Visualizaciones Clave

El bloque crea varios tipos de gráficos:

1. Gráficos de Barras

- Comparan churn entre diferentes categorías
- **Ejemplo**: Churn por tipo de contrato

2. Histogramas

- Muestran distribuciones de variables numéricas
- **Ejemplo**: Distribución de tenure para clientes que se van vs. se quedan

3. Box Plots (Diagramas de Caja)

- Muestran la distribución, mediana y valores atípicos
- **Ejemplo**: MonthlyCharges para cada grupo de churn

4. Heatmap de Correlación

- Matriz de colores que muestra correlaciones
- Colores cálidos (rojo) = correlación alta
- Colores fríos (azul) = correlación baja

Analogía: Es como un mapa de calor que muestra qué variables están "conectadas".

🔗 Relación con el Análisis General

El EDA es **fundamental** porque:

1. **Genera hipótesis**: Descubrimos que contratos largos reducen churn
2. **Identifica variables importantes**: Tenure, MonthlyCharges, Contract son clave
3. **Detecta problemas**: Desbalanceo de clases que necesitaremos manejar
4. **Informa decisiones**: Qué variables incluir en el modelo
5. **Comunica insights**: Las visualizaciones cuentan la historia a stakeholders

---

💡 Puntos Clave para Recordar

1. **27% de churn** - Problema significativo de negocio

2. **Desbalanceo de clases**: 73% No, 27% Yes
3. **Factores de riesgo de churn**:
    - Contratos mes a mes
    - Clientes nuevos (tenure bajo)
    - Cargos mensuales altos
    - Sin servicios adicionales (seguridad, soporte)
    - Fibra óptica (más cara)
4. **Factores protectores**:
    - Contratos largos (1-2 años)
    - Mayor antigüedad
    - Servicios adicionales contratados
5. **Correlaciones importantes**: Tenure y MonthlyCharges son predictores clave

---

🔬 Comprobación de Hipótesis Estadísticas

Después del análisis exploratorio visual, el notebook incluye **pruebas estadísticas formales** para validar las relaciones observadas.

¿Qué son las pruebas de hipótesis?

Analogía del juicio: En un juicio, no basta con "creer" que alguien es culpable. Necesitas **evidencia estadística** que demuestre la culpabilidad más allá de una duda razonable.

Las pruebas de hipótesis nos permiten determinar si las relaciones que observamos son **estadísticamente significativas** o podrían ser producto del azar.

Nivel de Significancia ($\alpha = 0.05$)

- Si **p-value < 0.05**: Rechazamos H_0 (hay evidencia estadística significativa)
- Si **p-value ≥ 0.05**: No rechazamos H_0 (no hay evidencia suficiente)

Analogía: Es como tener 95% de confianza de que algo es verdad, no solo una corazonada.

Pruebas Realizadas

El notebook incluye **7 pruebas de hipótesis**:

1. **Contract vs Churn** (Chi-cuadrado) - ✅ Significativa
2. **PaymentMethod vs Churn** (Chi-cuadrado) - ✅ Significativa
3. **InternetService vs Churn** (Chi-cuadrado) - ✅ Significativa
4. **tenure vs Churn** (Mann-Whitney U) - ✅ Significativa
5. **MonthlyCharges vs Churn** (Mann-Whitney U) - ✅ Significativa
6. **TechSupport vs Churn** (Chi-cuadrado) - ✅ Significativa
7. **PaperlessBilling vs Churn** (Chi-cuadrado) - ✅ Significativa

Implicaciones

Todas las variables mostraron **asociaciones estadísticamente significativas** con el churn (p-value < 0.05), lo que valida que:

- Las relaciones observadas en el EDA NO son casualidad
- Estas variables tienen poder predictivo real
- Están justificadas para incluirse en el modelo de ML

🎓 Conclusión

El EDA revela la **historia detrás de los números**: los clientes se van principalmente por precios altos y falta de compromiso (contratos cortos). Los clientes leales tienen contratos largos, servicios adicionales y llevan más tiempo con la empresa.

Las **pruebas de hipótesis estadísticas** confirman que estas relaciones NO son casualidad, sino que tienen **significancia estadística** (p-value < 0.05).

Estos insights no solo nos ayudan a construir mejores modelos, sino que también sugieren **estrategias de negocio**:

- Incentivar contratos largos
- Ofrecer descuentos en servicios adicionales
- Programas de retención para clientes nuevos
- Mejorar soporte técnico
- Revisar estrategia de facturación electrónica

Siguiente paso: Feature Engineering - crear nuevas variables basadas en estos insights validados estadísticamente.

Bloque 6: Feature Engineering

📋 Descripción General

Este bloque es como **un chef que combina ingredientes básicos para crear nuevos sabores**. Tomamos las variables existentes y creamos nuevas características (features) que pueden ayudar a los modelos a predecir mejor el churn.

🎯 Propósito y Objetivo

Los objetivos principales de este bloque son:

1. **Crear nuevas variables** derivadas de las existentes
2. **Capturar relaciones complejas** que no son obvias en los datos originales
3. **Mejorar el poder predictivo** del modelo
4. **Simplificar información** agrupando variables relacionadas

¿Por qué es importante?

Analogía del detective: Un detective no solo mira las pistas individuales, sino que las combina para formar una imagen completa. Por ejemplo:

- Pista 1: Huellas en la puerta
- Pista 2: Ventana rota
- **Nueva pista combinada**: Entrada forzada

De la misma manera, combinamos variables para crear información más útil.

🔑 Conceptos Clave y Técnicas Utilizadas

1. **Creación de Variables Derivadas**

El bloque crea varias nuevas características:

A) Promedio de Cargo Mensual por Mes de Antigüedad

```
df_fe['AvgChargesPerMonth'] = df_fe['TotalCharges'] / (df_fe['tenure'] + 1)
```

¿Qué mide?

- Cuánto paga el cliente en promedio por mes
- El +1 evita división por cero para clientes nuevos

¿Por qué es útil?

- Captura si el cliente ha tenido aumentos o descuentos a lo largo del tiempo
- Normaliza el gasto por la antigüedad

Analogía: Es como calcular tu gasto promedio mensual en café dividiendo lo que has gastado en total entre los meses que llevas comprando.

B) Número Total de Servicios Contratados

```
services = ['PhoneService', 'InternetService', 'OnlineSecurity',
            'OnlineBackup', 'DeviceProtection', 'TechSupport',
            'StreamingTV', 'StreamingMovies']
df_fe['TotalServices'] = (df_fe[services] != 'No').sum(axis=1)
```

¿Qué mide?

- Cuenta cuántos servicios tiene contratados el cliente (de 0 a 8)

¿Por qué es útil?

- El EDA mostró que clientes con más servicios tienen menos churn
- Resume 8 variables en una sola métrica

Analogía: Es como contar cuántos extras pediste en tu hamburguesa (queso, tocino, aguacate, etc.). Más extras = más comprometido con el restaurante.

C) Indicador de Cliente Premium

```
df_fe['IsPremium'] = ((df_fe['MonthlyCharges'] > df_fe['MonthlyCharges'].median()) &
                      (df_fe['TotalServices'] >= 3)).astype(int)
```

¿Qué mide?

- Si el cliente paga más que la mediana Y tiene 3+ servicios
- Valor: 1 (premium) o 0 (no premium)

¿Por qué es útil?

- Identifica clientes de alto valor
- Combina dos dimensiones: gasto y uso de servicios

Analogía: Como identificar clientes VIP en un hotel (gastan mucho Y usan muchos servicios).

D) Categorías de Antigüedad (Tenure Groups)

```
df_fe['TenureGroup'] = pd.cut(df_fe['tenure'],
                              bins=[0, 12, 24, 48, 72],
                              labels=['0-1 year', '1-2 years', '2-4 years', '4+ years'])
```

¿Qué hace pd.cut()?

- Divide una variable continua en categorías (bins)
- Como poner edades en grupos: niño, adolescente, adulto, anciano

Categorías creadas:

- **0-1 year**: Clientes nuevos (alto riesgo de churn)
- **1-2 years**: Clientes establecidos
- **2-4 years**: Clientes leales
- **4+ years**: Clientes muy leales (bajo riesgo)

¿Por qué es útil?

- Los modelos pueden capturar mejor relaciones no lineales
- Refleja que el riesgo de churn no disminuye uniformemente con el tiempo

Analogía: Como clasificar estudiantes por año (freshman, sophomore, junior, senior) en vez de solo por edad.

E) Indicador de Contrato Flexible

```
df_fe['HasFlexibleContract'] = (df_fe['Contract'] == 'Month-to-month').astype(int)
```

¿Qué mide?

- Si el cliente tiene contrato mes a mes (1) o no (0)

¿Por qué es útil?

- El EDA mostró que contratos mes a mes tienen ~42% de churn
- Simplifica la variable Contract en un indicador binario de riesgo

Analogía: Como marcar si alguien tiene un trabajo temporal vs. permanente.

F) Ratio de Servicios de Seguridad

```
security_services = ['OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport']
df_fe['SecurityServicesRatio'] = (df_fe[security_services] != 'No').sum(axis=1) /
len(security_services)
```

¿Qué mide?

- Proporción de servicios de seguridad contratados (0 a 1)
- 0 = ninguno, 0.5 = mitad, 1 = todos

¿Por qué es útil?

- Los servicios de seguridad están asociados con menor churn
- Normaliza el conteo en una escala de 0 a 1

Analogía: Como medir qué tan protegida está tu casa (alarma, cámaras, cerraduras, perro guardián) en una escala de 0% a 100%.

---

2. Transformaciones de Variables Existentes

Codificación de Variables Binarias

```
df_fe['gender'] = df_fe['gender'].map({'Male': 1, 'Female': 0})
df_fe['Partner'] = df_fe['Partner'].map({'Yes': 1, 'No': 0})
```

¿Por qué convertir a números?

- Los modelos de Machine Learning solo entienden números
- Yes/No → 1/0 es más eficiente que crear columnas dummy

📊 Resumen de Nuevas Features Creadas

Feature	Tipo	Descripción	Utilidad
AvgChargesPerMonth	Numérica	Cargo promedio por mes	Detecta cambios en precios
TotalServices	Numérica	Cantidad de servicios	Mide engagement del cliente
IsPremium	Binaria	Cliente de alto valor	Segmentación
TenureGroup	Categórica	Grupo de antigüedad	Captura no-linealidad
HasFlexibleContract	Binaria	Contrato mes a mes	Indicador de riesgo
SecurityServicesRatio	Numérica	Proporción de servicios de seguridad	Mide protección

🔗 Relación con el Análisis General

El Feature Engineering es el **puente entre el análisis y el modelado**:

1. **Usa insights del EDA**: Las features se basan en hallazgos del análisis exploratorio
2. **Prepara para el modelado**: Crea variables que los modelos pueden usar efectivamente
3. **Mejora el rendimiento**: Features bien diseñadas = mejores predicciones

4. **Reduce dimensionalidad**: Combina múltiples variables en métricas significativas

---

💡 Puntos Clave para Recordar

1. **Feature Engineering es un arte Y una ciencia**: Requiere creatividad y conocimiento del dominio
2. **Basado en insights**: Cada feature nueva debe tener una justificación lógica
3. **6 nuevas features** creadas a partir de las originales
4. **Combinación de enfoques**: Agregación, categorización, ratios, indicadores binarios
5. **Mejora interpretabilidad**: Features como `IsPremium` son fáciles de entender para el negocio

---

🎓 Conclusión

El Feature Engineering transforma datos crudos en información accionable. No solo creamos variables nuevas, sino que capturamos **conocimiento del negocio** en forma de features que los modelos pueden usar.

Ejemplo de impacto: En vez de que el modelo aprenda por sí solo que "contratos mes a mes + tenure bajo = alto riesgo", le damos directamente `HasFlexibleContract` y `TenureGroup` para facilitar su trabajo.

Siguiente paso: Preparar los datos para el modelado (división train/test, normalización, encoding).

Bloque 7: Preparación de Datos para Modelado

📋 Descripción General

Este bloque es como **preparar los ingredientes antes de cocinar**. Tenemos los datos limpios y las features creadas, pero ahora necesitamos transformarlos al formato exacto que los algoritmos de Machine Learning requieren.

🎯 Propósito y Objetivo

Los objetivos principales de este bloque son:

1. **Separar variables predictoras (X) de la variable objetivo (y)**
2. **Dividir datos en conjuntos de entrenamiento y prueba**
3. **Codificar variables categóricas** en formato numérico
4. **Normalizar variables numéricas** a la misma escala
5. **Crear un pipeline de preprocesamiento** automatizado

¿Por qué es importante?

Analogía del examen: Imagina que vas a tomar un examen: - **Entrenamiento**: Estudias con ejercicios de práctica - **Prueba**: Tomas el examen real con preguntas nuevas

Si estudias con las mismas preguntas del examen, memorizarás las respuestas pero no aprenderás realmente. Por eso separamos los datos.

🔑 Conceptos Clave y Técnicas Utilizadas

1. **Separación de Variables (X e y)**

```
X = df_model.drop(['Churn', 'customerID'], axis=1)
y = df_model['Churn'].map({'Yes': 1, 'No': 0})
```

¿Qué hace esto?

- **X**: Variables predictoras (features) - todo excepto Churn y customerID
- **y**: Variable objetivo - Churn convertido a 1 (Yes) y 0 (No)

¿Por qué eliminar customerID?

- Es solo un identificador único, no tiene poder predictivo
- Sería como usar el número de cédula para predecir si alguien se enferma

Analogía: X son las pistas que el detective tiene, y es el culpable que debe descubrir.

---

2. **División Train/Test (80/20)**

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state=42, stratify=y
)
```

Parámetros importantes:

test_size=0.20

- 80% de datos para entrenar
- 20% de datos para probar
- **Analogía**: De 100 problemas de matemáticas, practicas con 80 y te evalúan con 20 nuevos

random_state=42

- Fija la semilla aleatoria para reproducibilidad
- Siempre obtendremos la misma división
- **Analogía**: Como usar la misma baraja de cartas mezclada de la misma forma cada vez

stratify=y

- Mantiene la misma proporción de churn en train y test
- Si hay 27% de churn en total, habrá ~27% en train y ~27% en test
- **Analogía**: Si una clase tiene 30% niños y 70% niñas, al dividir en grupos mantienes esa proporción

¿Por qué es crítico?

- **Sin stratify**: Podrías tener 40% churn en train y 10% en test (desbalance)
- **Con stratify**: Ambos conjuntos son representativos

---

3. Identificación de Tipos de Variables

```
categorical_features = X_train.select_dtypes(include=['object']).columns.tolist()
numerical_features = X_train.select_dtypes(include=['int64', 'float64']).columns.tolist()
```

¿Por qué separar?

- Variables categóricas y numéricas requieren transformaciones diferentes
- **Categóricas**: Necesitan codificación (texto → números)
- **Numéricas**: Necesitan normalización (misma escala)

---

4. Codificación de Variables Categóricas (One-Hot Encoding)

```
OneHotEncoder(drop='first', sparse=False, handle_unknown='ignore')
```

¿Qué es One-Hot Encoding?

Convierte categorías en columnas binarias (0 o 1).

Ejemplo con InternetService:

- Original: [‘DSL’, ‘Fiber optic’, ‘No’]
- Después de One-Hot:
  - InternetService_Fiber optic: 1 si es Fiber, 0 si no
  - InternetService_No: 1 si es No, 0 si no
  - (DSL se infiere cuando ambas son 0)

Parámetros:

- **drop=‘first’**: Elimina la primera categoría para evitar multicolinealidad
- **sparse=False**: Retorna array denso (más fácil de manejar)
- **handle_unknown=‘ignore’**: Si aparece una categoría nueva en test, la ignora

Analogía: Es como tener casillas de verificación:

- ☐ DSL
- ☐ Fiber optic
- ☐ No internet

Marcas la que aplica (1) y dejas las demás vacías (0).

---

5. Normalización de Variables Numéricas (StandardScaler)

```
StandardScaler()
```

¿Qué hace StandardScaler?

Transforma los datos para que tengan:

- **Media = 0**
- **Desviación estándar = 1**

Fórmula: (valor - media) / desviación_estándar

¿Por qué es necesario?

Problema sin normalización:

- tenure: rango 0-72
- MonthlyCharges: rango 18-118
- TotalCharges: rango 0-8,000+

Algunos algoritmos (como SVM, KNN) son sensibles a la escala. Sin normalización, TotalCharges dominaría porque tiene valores mucho más grandes.

Analogía: Es como convertir todas las medidas a la misma unidad antes de compararlas:

- Altura: metros
- Peso: kilogramos
- Edad: años

Sin normalización sería como comparar metros con milímetros (los milímetros siempre parecerían más importantes por ser números más grandes).

---

6. Pipeline de Preprocesamiento

```
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_features),
        ('cat', OneHotEncoder(...), categorical_features)
    ]
)
```

¿Qué es un Pipeline?

Un flujo de trabajo automatizado que aplica transformaciones en orden.

Ventajas:

1. **Automatización**: Aplica todas las transformaciones con un solo comando
2. **Consistencia**: Las mismas transformaciones se aplican a train y test
3. **Previene data leakage**: No usa información de test para transformar train
4. **Reproducibilidad**: Fácil de replicar

Analogía: Es como una línea de ensamblaje en una fábrica:

- Estación 1: Normalizar números
- Estación 2: Codificar categorías
- Producto final: Datos listos para el modelo

---

7. Aplicación del Preprocesamiento

```
X_train_processed = preprocessor.fit_transform(X_train)
X_test_processed = preprocessor.transform(X_test)
```

Diferencia crítica:

- **fit_transform** en train: Aprende los parámetros (media, desviación) Y transforma
- **transform** en test: Solo transforma usando los parámetros aprendidos de train

¿Por qué esta diferencia?

Analogía del profesor:

- El profesor (preprocessor) aprende de los estudiantes de práctica (train)
- Luego aplica lo aprendido a los estudiantes del examen (test)
- NO debe aprender nada de los estudiantes del examen (evita data leakage)

📊 Resultado de la Preparación

Antes:

- Variables categóricas como texto
- Variables numéricas en diferentes escalas
- Todo en un solo DataFrame

Después:

- Todo convertido a números
- Variables normalizadas (media=0, std=1)
- Listo para alimentar a los modelos

Dimensiones:

- **X_train**: ~5,634 filas (80%)
- **X_test**: ~1,409 filas (20%)
- **Columnas**: Aumentan por One-Hot Encoding

🔗 Relación con el Análisis General

Este bloque es el **último paso antes del modelado**:

1. **Cierra el preprocesamiento**: Datos completamente listos
2. **Previene errores comunes**: Data leakage, escalas incorrectas
3. **Optimiza el rendimiento**: Modelos funcionan mejor con datos normalizados
4. **Facilita la evaluación**: División train/test permite medir rendimiento real

---

💡 Puntos Clave para Recordar

1. **División 80/20** con stratify para mantener proporciones
2. **One-Hot Encoding** para variables categóricas
3. **StandardScaler** para variables numéricas
4. **Pipeline** automatiza y asegura consistencia
5. **fit_transform** en train, **transform** en test (evita data leakage)
6. **random_state=42** para reproducibilidad

🎓 Conclusión

La preparación de datos es como preparar el escenario antes de una obra de teatro: todo debe estar en su lugar, con el formato correcto y listo para la acción.

Un preprocesamiento adecuado es la diferencia entre un modelo que funciona y uno que falla.

Siguiente paso: Entrenar múltiples modelos baseline y comparar su rendimiento.

Bloque 8: Entrenamiento de Modelos Baseline

📋 Descripción General

Este bloque es como **una competencia deportiva donde varios atletas compiten** para ver quién es el mejor. Entrenamos múltiples algoritmos de Machine Learning diferentes y comparamos su rendimiento para identificar cuáles funcionan mejor para predecir el churn.

🎯 Propósito y Objetivo

Los objetivos principales de este bloque son:

1. **Entrenar múltiples modelos** con configuraciones por defecto (baseline)
2. **Evaluar el rendimiento** de cada modelo con métricas apropiadas
3. **Comparar resultados** para identificar los mejores candidatos
4. **Establecer una línea base** de rendimiento antes de optimizar

¿Por qué probar múltiples modelos?

Analogía del transporte: Si necesitas ir de A a B, podrías usar:

- Bicicleta (rápida para distancias cortas)
- Auto (versátil)
- Tren (eficiente para largas distancias)
- Avión (rápido pero costoso)

Cada uno tiene ventajas y desventajas. Lo mismo pasa con los algoritmos: cada uno tiene fortalezas en diferentes tipos de problemas.

🔑 Modelos Entrenados y Sus Características

1. **Logistic Regression (Regresión Logística)**

¿Cómo funciona?

- Encuentra una línea (o hiperplano) que separa las dos clases
- Calcula la probabilidad de que un cliente haga churn

Ventajas:

- ✅ Simple y rápido
- ✅ Fácil de interpretar
- ✅ Funciona bien con relaciones lineales

Desventajas:

- ❌ Asume relaciones lineales
- ❌ No captura patrones complejos

Analogía: Es como trazar una línea recta en un mapa para separar dos regiones.

---

2. Decision Tree (Árbol de Decisión)

¿Cómo funciona?

- Hace una serie de preguntas (if-then-else)
- Cada pregunta divide los datos en grupos más puros

Ejemplo de decisiones:

```
¿Contrato mes a mes?
├─ Sí → ¿Tenure < 12 meses?
│  ├─ Sí → CHURN (alta probabilidad)
│  └─ No → NO CHURN
└─ No → NO CHURN
```

Ventajas:

- ✅ Muy interpretable
- ✅ Captura relaciones no lineales
- ✅ No requiere normalización

Desventajas:

- ❌ Propenso a overfitting (memorizar en vez de aprender)
- ❌ Inestable (pequeños cambios en datos → árbol muy diferente)

Analogía: Como un diagrama de flujo de decisiones que sigues paso a paso.

---

3. Random Forest (Bosque Aleatorio)

¿Cómo funciona?

- Crea muchos árboles de decisión (100-1000)
- Cada árbol vota
- La decisión final es por mayoría

Ventajas:

- ✅ Muy robusto y preciso
- ✅ Reduce overfitting vs. un solo árbol
- ✅ Maneja bien datos complejos
- ✅ Proporciona importancia de features

Desventajas:

- ❌ Menos interpretable que un solo árbol
- ❌ Más lento de entrenar

Analogía: Como pedir opinión a 100 expertos y tomar la decisión por votación mayoritaria.

---

4. Gradient Boosting

¿Cómo funciona?

- Construye árboles secuencialmente
- Cada árbol nuevo corrige los errores del anterior
- Es como aprender de tus errores iterativamente

Ventajas:

- ✅ Muy preciso
- ✅ Captura patrones complejos
- ✅ Funciona bien en competencias de ML

Desventajas:

- ❌ Más lento de entrenar
- ❌ Requiere ajuste cuidadoso de parámetros
- ❌ Propenso a overfitting si no se configura bien

Analogía: Como un estudiante que hace un examen de práctica, revisa sus errores, estudia esas áreas y mejora en el siguiente intento.

---

5. XGBoost (Extreme Gradient Boosting)

¿Cómo funciona?

- Versión optimizada y mejorada de Gradient Boosting
- Incluye regularización para prevenir overfitting
- Muy eficiente computacionalmente

Ventajas:

- ✅ Estado del arte en muchos problemas
- ✅ Muy preciso
- ✅ Maneja bien datos desbalanceados
- ✅ Rápido (comparado con Gradient Boosting tradicional)

Desventajas:

- ❌ Muchos hiperparámetros para ajustar
- ❌ Menos interpretable

Analogía: Como Gradient Boosting pero con turbo y mejor motor.

---

6. Support Vector Machine (SVM)

¿Cómo funciona?

- Encuentra el mejor hiperplano que separa las clases
- Maximiza el margen entre las clases

Ventajas:

- ✅ Efectivo en espacios de alta dimensión
- ✅ Funciona bien con datos no lineales (usando kernels)

Desventajas:

- ❌ Lento con datasets grandes
- ❌ Sensible a la escala de datos
- ❌ Difícil de interpretar

Analogía: Como encontrar la mejor valla para separar dos rebaños de ovejas, maximizando el espacio entre ellas.

---

7. K-Nearest Neighbors (KNN)

¿Cómo funciona?

- Para clasificar un punto, mira sus K vecinos más cercanos
- Asigna la clase más común entre esos vecinos

Ventajas:

- ✅ Simple de entender
- ✅ No requiere entrenamiento (lazy learning)

Desventajas:

- ❌ Lento en predicción con datasets grandes
- ❌ Sensible a la escala y ruido
- ❌ Requiere elegir K apropiado

Analogía: "Dime con quién andas y te diré quién eres" - si tus 5 vecinos más cercanos hicieron churn, probablemente tú también.

📊 Métricas de Evaluación

El bloque evalúa cada modelo con múltiples métricas:

1. Accuracy (Exactitud)

- **¿Qué mide?** Porcentaje de predicciones correctas
- **Fórmula**: (Aciertos totales) / (Total de predicciones)
- **Problema**: Puede ser engañosa con datos desbalanceados

Ejemplo: Si 73% de clientes NO hacen churn, un modelo que siempre predice "NO" tendría 73% de accuracy pero sería inútil.

2. Precision (Precisión)

- **¿Qué mide?** De los que predijimos como churn, ¿cuántos realmente lo hicieron?
- **Fórmula**: Verdaderos Positivos / (Verdaderos Positivos + Falsos Positivos)
- **Importancia**: Evita falsas alarmas

Analogía: De todas las veces que el detector de humo sonó, ¿cuántas veces realmente había fuego?

3. Recall (Sensibilidad)

- **¿Qué mide?** De todos los que realmente hicieron churn, ¿cuántos detectamos?

- **Fórmula**: Verdaderos Positivos / (Verdaderos Positivos + Falsos Negativos)
- **Importancia**: No perder clientes en riesgo

Analogía: De todos los incendios que hubo, ¿cuántos detectó el detector de humo?

4. F1-Score

- **¿Qué mide?** Balance entre Precision y Recall
- **Fórmula**: 2 × (Precision × Recall) / (Precision + Recall)
- **Importancia**: Métrica equilibrada

5. ROC-AUC

- **¿Qué mide?** Capacidad del modelo para discriminar entre clases
- **Rango**: 0.5 (aleatorio) a 1.0 (perfecto)
- **Importancia**: Independiente del umbral de decisión

🏆 Resultados Típicos (Baseline)

Modelos de mejor rendimiento (generalmente):

1. **XGBoost**: ~85% accuracy, ~0.85 ROC-AUC
2. **Random Forest**: ~84% accuracy, ~0.84 ROC-AUC
3. **Gradient Boosting**: ~83% accuracy, ~0.83 ROC-AUC

Modelos de rendimiento moderado:

4. **Logistic Regression**: ~80% accuracy
5. **SVM**: ~79% accuracy

Modelos de menor rendimiento:

6. **Decision Tree**: ~75% accuracy (overfitting)
7. **KNN**: ~76% accuracy

🔗 Relación con el Análisis General

Este bloque es **crucial** porque:

1. **Identifica candidatos**: Descubrimos qué modelos funcionan mejor
2. **Establece baseline**: Punto de referencia para mejoras futuras
3. **Informa optimización**: Sabemos en qué modelos invertir tiempo
4. **Valida el enfoque**: Confirma que el problema es predecible

---

💡 Puntos Clave para Recordar

1. **7 modelos diferentes** entrenados y comparados
2. **Ensemble methods** (Random Forest, Gradient Boosting, XGBoost) suelen ganar
3. **Accuracy no es suficiente** - necesitamos múltiples métricas
4. **Baseline** = **configuración por defecto** - aún no optimizado
5. **Desbalanceo de clases** afecta el rendimiento (se abordará en el siguiente bloque)

🎓 Conclusión

El entrenamiento de modelos baseline es como una audición: probamos varios candidatos para ver quiénes tienen potencial. Los modelos ensemble (Random Forest, XGBoost) generalmente destacan, pero todos aportan información valiosa.

Siguiente paso: Manejar el desbalanceo de clases con técnicas como SMOTE para mejorar la detección de churn.

Bloque 9: Manejo del Desbalanceo de Clases

📋 Descripción General

Este bloque es como **equilibrar una balanza desnivelada**. Recordemos que tenemos 73% de clientes que NO hacen churn y solo 27% que SÍ lo hacen. Este desbalanceo puede hacer que los modelos sean "perezosos" y simplemente predigan siempre la clase mayoritaria. Aquí aplicamos técnicas para balancear las clases y mejorar la detección de churn.

🎯 Propósito y Objetivo

Los objetivos principales de este bloque son:

1. **Aplicar SMOTE** para balancear las clases en el conjunto de entrenamiento
2. **Reentrenar los mejores modelos** con datos balanceados
3. **Comparar resultados** antes y después del balanceo
4. **Mejorar el Recall** (detección de clientes que harán churn)

¿Por qué es importante?

Analogía de la enfermedad rara: Imagina un test médico para una enfermedad que solo afecta al 3% de la población:

- Un modelo "tonto" que siempre dice "NO tienes la enfermedad" tendría 97% de accuracy
- Pero sería inútil porque nunca detectaría a los enfermos

Lo mismo pasa con el churn: necesitamos detectar específicamente a los que SÍ se van, no solo tener alta accuracy general.

🔑 Conceptos Clave y Técnicas Utilizadas

1. **El Problema del Desbalanceo**

Distribución original:

- **No Churn**: ~5,163 clientes (73%)
- **Churn**: ~1,869 clientes (27%)

Ratio: ~2.76:1 (casi 3 veces más "No" que "Yes")

Consecuencias:

- Los modelos aprenden mejor la clase mayoritaria
- Baja sensibilidad para detectar churn
- Métricas engañosas (alta accuracy pero bajo recall)

Analogía del profesor: Si en una clase hay 73 estudiantes callados y 27 habladores, el profesor prestará más atención a los callados (mayoría) y puede ignorar a los habladores (minoría).

---

2. SMOTE (Synthetic Minority Over-sampling Technique)

¿Qué es SMOTE?

Una técnica que crea **ejemplos sintéticos** de la clase minoritaria (Churn=Yes) para balancear el dataset.

¿Cómo funciona?

1. Toma un ejemplo de la clase minoritaria
2. Encuentra sus K vecinos más cercanos (también de la clase minoritaria)
3. Crea nuevos ejemplos interpolando entre el ejemplo original y sus vecinos

Analogía visual:

```
Original:  A -------- B
              ↓
Sintético: A -- X -- B
```

Donde X es un nuevo ejemplo creado "entre" A y B.

Ventajas de SMOTE:

- ✅ Crea ejemplos realistas (no duplicados)
- ✅ Aumenta la diversidad de la clase minoritaria
- ✅ Mejora el aprendizaje del modelo

Diferencia con otras técnicas:

- **RandomOverSampler**: Simplemente duplica ejemplos existentes (puede causar overfitting)
- **RandomUnderSampler**: Elimina ejemplos de la clase mayoritaria (pierde información)
- **SMOTE**: Crea nuevos ejemplos sintéticos (balance sin pérdida de información)

---

3. Aplicación de SMOTE

```
smote = SMOTE(random_state=42)
X_train_balanced, y_train_balanced = smote.fit_resample(X_train, y_train)
```

Resultado:

- **Antes**: 5,163 No Churn, 1,869 Churn
- **Después**: 5,163 No Churn, 5,163 Churn (balanceado 50/50)

Importante: SMOTE solo se aplica al conjunto de **entrenamiento**, NUNCA al de prueba.

¿Por qué?

Analogía del examen:

- Puedes estudiar con material adicional (SMOTE en train)
- Pero el examen debe ser con preguntas reales (test sin modificar)

---

4. Reentrenamiento de Modelos

El bloque reentrena los mejores modelos (identificados en el bloque anterior) con los datos balanceados:

1. **Logistic Regression**
2. **Random Forest**
3. **Gradient Boosting**
4. **XGBoost**

Configuración: Se mantienen los parámetros por defecto (baseline) para comparación justa.

📊 Comparación de Resultados: Antes vs. Después de SMOTE

Métricas Típicas - Antes de SMOTE

Modelo	Accuracy	Precision	Recall	F1-Score
Random Forest	84%	70%	50%	58%
XGBoost	85%	72%	52%	60%

Problema: Recall bajo (solo detecta ~50% de los churns)

Métricas Típicas - Después de SMOTE

Modelo	Accuracy	Precision	Recall	F1-Score
Random Forest	82%	65%	78%	71%
XGBoost	83%	67%	80%	73%

Mejora: Recall aumenta significativamente (~30% de mejora)

Interpretación de los Cambios

Accuracy baja ligeramente (84% → 82%)

- **¿Por qué?** Ahora el modelo comete más “falsos positivos” (predice churn cuando no lo hay)
- **¿Es malo?** No necesariamente - depende del objetivo de negocio

Recall aumenta significativamente (50% → 78%)

- **¿Por qué?** El modelo ahora detecta mejor la clase minoritaria
- **¿Es bueno?** ¡Sí! Detectamos más clientes en riesgo

***Precision baja un poco** (70% → 65%)*

- **¿Por qué?** Más falsos positivos
- **Trade-off**: Sacrificamos un poco de precisión por mucho más recall

---

Trade-off: Precision vs. Recall

Analogía del detector de metales:

Antes de SMOTE (Alta Precision, Bajo Recall):

- Cuando pita, casi siempre hay metal (pocas falsas alarmas)
- Pero se pierde mucho metal (no detecta todo)

Después de SMOTE (Precision moderada, Alto Recall):

- Pita más veces, algunas falsas alarmas
- Pero encuentra casi todo el metal

Para el negocio de Telco:

- **Falso Positivo**: Ofrecemos descuento a alguien que no se iba a ir (costo: descuento innecesario)
- **Falso Negativo**: No detectamos a alguien que se va (costo: perder el cliente completo)

Conclusión: Es mejor tener algunos falsos positivos que perder clientes reales.

🔗 Relación con el Análisis General

Este bloque es **crítico** porque:

1. **Corrige un problema fundamental**: El desbalanceo de clases
2. **Mejora la métrica clave**: Recall (detección de churn)
3. **Alinea con objetivos de negocio**: Preferimos detectar más churns aunque tengamos algunas falsas alarmas
4. **Prepara para optimización**: Datos balanceados permiten mejor ajuste de hiperparámetros

---

💡 Puntos Clave para Recordar

1. **Desbalanceo original**: 73% No Churn, 27% Churn
2. **SMOTE** crea ejemplos sintéticos de la clase minoritaria
3. **Solo se aplica a train**, nunca a test
4. **Recall mejora ~30%** (de ~50% a ~78%)
5. **Trade-off**: Accuracy baja un poco, pero Recall aumenta mucho
6. **Para el negocio**: Mejor detectar más churns con algunas falsas alarmas

---

🎓 Conclusión

El manejo del desbalanceo de clases transforma un modelo “perezoso” que ignora la clase minoritaria en uno que realmente detecta clientes en riesgo. SMOTE es como darle al modelo “gafas especiales” para ver mejor la clase minoritaria.

Lección importante: En problemas de negocio, la métrica más importante no siempre es accuracy. Para churn, Recall es crítico porque el costo de perder un cliente es mucho mayor que el costo de una falsa alarma.

Siguiente paso: Optimizar hiperparámetros de los mejores modelos para exprimir el máximo rendimiento.

Bloque 10: Optimización de Hiperparámetros

📋 Descripción General

Este bloque es como **afinar un instrumento musical** para que suene perfecto. Los modelos tienen "perillas" (hiperparámetros) que controlan su comportamiento. Aquí buscamos la mejor combinación de configuraciones para maximizar el rendimiento del modelo.

🎯 Propósito y Objetivo

Los objetivos principales de este bloque son:

1. **Identificar el mejor modelo** de los candidatos anteriores
2. **Definir un espacio de búsqueda** de hiperparámetros
3. **Aplicar GridSearchCV o RandomizedSearchCV** para encontrar la mejor configuración
4. **Evaluar el modelo optimizado** y comparar con el baseline
5. **Seleccionar el modelo final** para producción

¿Por qué es importante?

Analogía del café: Hacer café perfecto requiere ajustar:

- Temperatura del agua
- Tiempo de extracción
- Cantidad de café
- Molienda

Cambiar cualquiera de estos parámetros afecta el sabor final. Lo mismo pasa con los modelos de ML.

🔑 Conceptos Clave y Técnicas Utilizadas

1. **¿Qué son los Hiperparámetros?**

Hiperparámetros son configuraciones que se establecen ANTES del entrenamiento y controlan cómo aprende el modelo.

Diferencia con parámetros:

- **Parámetros**: El modelo los aprende de los datos (ej: pesos en regresión)
- **Hiperparámetros**: Los definimos nosotros (ej: profundidad de un árbol)

Analogía del estudiante:

- **Parámetros**: El conocimiento que adquiere estudiando
- **Hiperparámetros**: Cuántas horas estudia, qué técnica usa, en qué ambiente

---

2. Hiperparámetros Comunes por Modelo

Random Forest

```
param_grid = {
    'n_estimators': [100, 200, 300],      # Número de árboles
    'max_depth': [10, 20, 30, None],      # Profundidad máxima
    'min_samples_split': [2, 5, 10],      # Mínimo para dividir nodo
    'min_samples_leaf': [1, 2, 4],        # Mínimo en hoja
    'max_features': ['sqrt', 'log2']      # Features por split
}
```

Explicación:

- **n_estimators**: Más árboles = más robusto pero más lento
  - **Analogía**: Más jueces en un panel = decisión más confiable
- **max_depth**: Controla cuán profundo puede crecer cada árbol
  - **Analogía**: Cuántos niveles de preguntas puede hacer
  - Muy profundo = overfitting, muy superficial = underfitting
- **min_samples_split**: Mínimo de ejemplos para dividir un nodo
  - **Analogía**: No dividir un grupo si es muy pequeño
  - Previene overfitting
- **min_samples_leaf**: Mínimo de ejemplos en cada hoja
  - **Analogía**: Cada conclusión debe basarse en al menos X casos
- **max_features**: Cuántas features considerar en cada split
  - **sqrt**: Raíz cuadrada del total (más diversidad)
  - **log2**: Logaritmo base 2 (más conservador)

XGBoost

```
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1, 0.3],
    'subsample': [0.8, 0.9, 1.0],
    'colsample_bytree': [0.8, 0.9, 1.0],
    'gamma': [0, 0.1, 0.2]
}
```

Explicación:

- **learning_rate**: Qué tan rápido aprende el modelo
  - **Analogía**: Tamaño del paso al caminar
  - Bajo (0.01) = lento pero preciso
  - Alto (0.3) = rápido pero puede pasarse
- **subsample**: Fracción de datos usados en cada árbol
  - **Analogía**: Usar una muestra aleatoria para cada decisión
  - Previene overfitting
- **colsample_bytree**: Fracción de features usadas por árbol
  - Similar a max_features en Random Forest

- **gamma**: Reducción mínima de pérdida para hacer un split
    - **Analogía**: Cuánto debe mejorar una pregunta para hacerla
    - Mayor gamma = modelo más conservador

---

3. GridSearchCV vs. RandomizedSearchCV

GridSearchCV (Búsqueda Exhaustiva)

¿Cómo funciona? - Prueba TODAS las combinaciones posibles de hiperparámetros

Ejemplo:

```
n_estimators: [100, 200]
max_depth: [10, 20]
```

Combinaciones: 2 × 2 = 4 pruebas

Ventajas:

- ✅ Garantiza encontrar la mejor combinación en el espacio definido
- ✅ Exhaustivo

Desventajas:

- ❌ Muy lento con muchos parámetros
- ❌ Crece exponencialmente

Analogía: Probar todas las combinaciones de ropa en tu armario.

RandomizedSearchCV (Búsqueda Aleatoria)

¿Cómo funciona?

- Prueba un número fijo de combinaciones aleatorias

Ejemplo:

```
n_estimators: [100, 200, 300, 400, 500]
max_depth: [5, 10, 15, 20, 25, 30]
learning_rate: [0.01, 0.05, 0.1, 0.15, 0.2]
```

Combinaciones posibles: 5 × 6 × 5 = 150 Pero solo prueba, por ejemplo, 30 aleatorias

Ventajas:

- ✅ Mucho más rápido
- ✅ Puede explorar espacios grandes
- ✅ Sorprendentemente efectivo

Desventajas:

- ❌ No garantiza encontrar el óptimo absoluto
- ❌ Resultados pueden variar entre ejecuciones

Analogía: Probar 20 combinaciones aleatorias de ropa en vez de todas.

---

4. Validación Cruzada (Cross-Validation)

Tanto GridSearchCV como RandomizedSearchCV usan **validación cruzada** para evaluar cada combinación.

¿Qué es CV=5?

Divide los datos de entrenamiento en 5 partes (folds):

```
Fold 1: [Test] [Train] [Train] [Train] [Train]
Fold 2: [Train] [Test] [Train] [Train] [Train]
Fold 3: [Train] [Train] [Test] [Train] [Train]
Fold 4: [Train] [Train] [Train] [Test] [Train]
Fold 5: [Train] [Train] [Train] [Train] [Test]
```

Proceso:

1. Entrena con 4 folds, evalúa en 1
2. Repite 5 veces (cada fold es test una vez)
3. Promedia los resultados

Ventajas:

- ✅ Usa todos los datos para entrenar y evaluar
- ✅ Resultados más robustos
- ✅ Reduce varianza de la evaluación

Analogía: En vez de un solo examen, tomas 5 exámenes diferentes y promedias la nota.

---

5. Proceso de Optimización

```
grid_search = GridSearchCV(
    estimator=RandomForestClassifier(random_state=42),
    param_grid=param_grid,
    cv=5,
    scoring='f1',
    n_jobs=-1,
    verbose=2
)

grid_search.fit(X_train_balanced, y_train_balanced)
best_model = grid_search.best_estimator_
```

Parámetros importantes:

- **estimator**: El modelo a optimizar
- **param_grid**: Espacio de búsqueda
- **cv=5**: 5-fold cross-validation
- **scoring='f1'**: Métrica a optimizar (F1-Score)
- **n_jobs=-1**: Usa todos los cores del CPU (paralelización)
- **verbose=2**: Muestra progreso detallado

Resultado:

- **best_estimator_**: Modelo con los mejores hiperparámetros
- **best_params_**: Diccionario con la mejor configuración

- **best_score_**: Mejor score obtenido en CV

📊 Resultados Típicos de la Optimización

Antes de Optimización (Baseline con SMOTE)

Modelo	F1-Score	Recall	Precision
Random Forest	0.71	0.78	0.65
XGBoost	0.73	0.80	0.67

Después de Optimización

Modelo	F1-Score	Recall	Precision
Random Forest	0.75	0.82	0.69
XGBoost	0.77	0.84	0.71

Mejora: ~4-5% en todas las métricas

Mejores Hiperparámetros Encontrados (Ejemplo)

XGBoost:

```
{
   'n_estimators': 200,
   'max_depth': 5,
   'learning_rate': 0.1,
   'subsample': 0.9,
   'colsample_bytree': 0.9,
   'gamma': 0.1
}
```

Interpretación:

- 200 árboles (balance entre rendimiento y velocidad)
- Profundidad moderada (5) para evitar overfitting
- Learning rate moderado (0.1) para convergencia estable
- Subsample y colsample altos (0.9) para usar la mayoría de datos/features
- Gamma bajo (0.1) para permitir splits útiles

🔗 Relación con el Análisis General

Este bloque es el **refinamiento final**:

1. **Maximiza el rendimiento**: Exprime el último % de mejora
2. **Selecciona el modelo final**: El que irá a producción
3. **Documenta la configuración**: Reproducibilidad
4. **Valida robustez**: CV asegura que no es suerte

💡 Puntos Clave para Recordar

1. **Hiperparámetros** controlan cómo aprende el modelo
2. **GridSearchCV**: Exhaustivo pero lento

3. **RandomizedSearchCV**: Rápido y efectivo para espacios grandes
4. **Cross-Validation (CV=5)**: Evaluación robusta
5. **Mejora típica**: 3-5% sobre baseline
6. **Scoring='f1'**: Optimizamos F1-Score (balance precision/recall)

🎓 Conclusión

La optimización de hiperparámetros es como ajustar la receta perfecta: pequeños cambios en los ingredientes pueden hacer una gran diferencia. No siempre da mejoras dramáticas, pero ese 3-5% extra puede ser la diferencia entre un modelo bueno y uno excelente.

Siguiente paso: Evaluación detallada del mejor modelo con análisis de errores, curvas ROC, feature importance y recomendaciones de negocio.

Bloque 11: Evaluación Detallada del Mejor Modelo

📋 Descripción General

Este bloque es como **el informe final de un proyecto de investigación**. Después de entrenar, balancear y optimizar múltiples modelos, ahora evaluamos exhaustivamente el mejor modelo seleccionado, analizamos sus fortalezas y debilidades, y generamos recomendaciones accionables para el negocio.

🎯 Propósito y Objetivo

Los objetivos principales de este bloque son:

1. **Evaluar el modelo final** con el conjunto de prueba (test set)
2. **Analizar la matriz de confusión** para entender tipos de errores
3. **Generar curvas ROC y Precision-Recall** para visualizar rendimiento
4. **Identificar feature importance** (variables más importantes)
5. **Analizar errores** para entender dónde falla el modelo
6. **Generar recomendaciones de negocio** basadas en los hallazgos

¿Por qué es importante?

Analogía del médico: Después de desarrollar un nuevo tratamiento:

- No basta con decir "funciona en el 85% de casos"
- Necesitas saber: ¿En qué casos falla? ¿Por qué? ¿Cómo mejorarlo?
- ¿Qué efectos secundarios tiene?

🔑 Conceptos Clave y Análisis Realizados

1. **Evaluación en el Conjunto de Prueba**

Importante: Hasta ahora, todas las optimizaciones se hicieron con datos de entrenamiento (usando CV). Ahora evaluamos con datos que el modelo NUNCA ha visto.

```
y_pred = best_model.predict(X_test)
y_pred_proba = best_model.predict_proba(X_test)[:, 1]
```

Métricas finales típicas:

- **Accuracy**: ~83%
- **Precision**: ~70%
- **Recall**: ~82%
- **F1-Score**: ~76%
- **ROC-AUC**: ~0.87

Interpretación: El modelo detecta correctamente ~82% de los clientes que harán churn.

---

2. **Matriz de Confusión**

La matriz de confusión muestra los 4 tipos de predicciones:

```
                Predicción
              No Churn  |  Churn
            +-----------+--------+
Real No Churn |    950    |   80   |  = 1030 (Verdaderos Negativos + Falsos Positivos)
            +-----------+--------+
Real Churn    |    68     |  311   |  = 379 (Falsos Negativos + Verdaderos Positivos)
            +-----------+--------+
```

Desglose:

Verdaderos Negativos (TN): 950

- Predijimos “No Churn” y era correcto
- ✅ Clientes leales correctamente identificados

Verdaderos Positivos (TP): 311

- Predijimos “Churn” y era correcto
- ✅ Clientes en riesgo correctamente detectados
- **Acción**: Ofrecer incentivos de retención

Falsos Positivos (FP): 80

- Predijimos “Churn” pero NO se fueron
- ⚠️ Falsa alarma
- **Costo**: Ofrecer descuentos innecesarios
- **Impacto**: Bajo (mejor prevenir que lamentar)

Falsos Negativos (FN): 68

- Predijimos “No Churn” pero SÍ se fueron
- ❌ Clientes en riesgo que NO detectamos
- **Costo**: Perder el cliente completo
- **Impacto**: Alto (pérdida de ingresos)

Análisis de Costos de Negocio

Supuestos:

- Costo de retención (descuento/incentivo): $50
- Valor de vida del cliente (CLV): $1,500
- Costo de perder un cliente: $1,500

Cálculo de costos:

1. **Falsos Positivos (80 clientes)**:
    - Costo: 80 × $50 = $4,000
    - (Ofrecemos descuentos innecesarios)
2. **Falsos Negativos (68 clientes)**:
    - Costo: 68 × $1,500 = $102,000
    - (Perdemos clientes que no detectamos)
3. **Verdaderos Positivos (311 clientes)**:
    - Inversión: 311 × $50 = $15,550
    - Ahorro (si retenemos 70%): 218 × $1,500 = $327,000
    - **Beneficio neto**: $327,000 - $15,550 = $311,450

Total:

- **Costos**: $4,000 + $102,000 + $15,550 = $121,550
- **Beneficios**: $327,000
- **ROI**: ~$205,000 de beneficio neto

Conclusión: El modelo es altamente rentable para el negocio.

---

3. Curva ROC (Receiver Operating Characteristic)

¿Qué es?

- Gráfico que muestra el trade-off entre True Positive Rate (Recall) y False Positive Rate
- Eje Y: Recall (sensibilidad)
- Eje X: False Positive Rate (1 - especificidad)

Interpretación del AUC (Area Under Curve):

- **AUC** = **0.50**: Modelo aleatorio (inútil)
- **AUC** = **0.70-0.80**: Aceptable
- **AUC** = **0.80-0.90**: Excelente
- **AUC** = **0.90-1.00**: Sobresaliente
- **AUC** = **1.00**: Perfecto (sospechoso de overfitting)

Nuestro modelo: AUC ~0.87 (Excelente)

Analogía: Es como medir qué tan bien un detector de metales distingue entre metal y no-metal en diferentes sensibilidades.

---

4. Curva Precision-Recall

¿Cuándo es más útil que ROC?

- Con datos desbalanceados (como nuestro caso: 27% churn)
- Cuando los Falsos Negativos son muy costosos

Interpretación:

- Muestra el trade-off entre Precision y Recall
- Permite elegir el umbral óptimo según prioridades de negocio

Ejemplo de umbrales:

Umbral	Precision	Recall	Uso
0.3	60%	90%	Campaña agresiva (detectar todos los riesgos)
0.5	70%	82%	Balance (configuración actual)
0.7	85%	65%	Campaña conservadora (solo casos muy seguros)

Recomendación: Usar umbral 0.4-0.5 para maximizar Recall sin sacrificar mucho Precision.

---

5. Feature Importance (Importancia de Variables)

El modelo identifica qué variables son más importantes para predecir churn:

Top 10 Features más importantes (ejemplo):

1. **tenure** (Antigüedad): 18%
    - Clientes nuevos tienen mucho más riesgo
2. **MonthlyCharges** (Cargo mensual): 15%
    - Precios altos aumentan churn
3. **Contract_Month-to-month**: 12%
    - Contratos flexibles = alto riesgo
4. **TotalCharges** (Cargo total): 10%
    - Relacionado con tenure
5. **InternetService_Fiber optic**: 8%
    - Fibra óptica (más cara) = más churn
6. **OnlineSecurity_No**: 7%
    - Sin servicios de seguridad = más riesgo
7. **TechSupport_No**: 6%
    - Sin soporte técnico = más riesgo
8. **PaymentMethod_Electronic check**: 5%
    - Método de pago menos comprometido

9. **PaperlessBilling_Yes**: 4%
    - Facturación sin papel = menos engagement
10. **SeniorCitizen**: 3%
    - Adultos mayores = más riesgo

Insights de Feature Importance

Factores de riesgo principales:

1. **Compromiso bajo**: Contratos cortos, tenure bajo
2. **Precio alto**: MonthlyCharges elevados
3. **Servicios limitados**: Sin seguridad, sin soporte
4. **Tipo de servicio**: Fibra óptica (premium)

Analogía: Es como descubrir que los estudiantes que faltan mucho (tenure bajo), no participan en actividades (sin servicios), y pagan más (MonthlyCharges altos) son los que más probablemente abandonan la escuela.

---

6. Análisis de Errores

Perfil de Falsos Negativos (clientes que se fueron pero no detectamos):

Características comunes:

- Tenure entre 12-24 meses (ni muy nuevos ni muy antiguos)
- MonthlyCharges moderados ($60-$80)
- Tienen algunos servicios adicionales
- Contratos de 1 año (no mes a mes)

Hipótesis: Estos clientes están en una "zona gris" donde el modelo tiene menos confianza.

Perfil de Falsos Positivos (predijimos churn pero se quedaron):

Características comunes:

- Tenure bajo (<6 meses) pero se quedaron
- MonthlyCharges altos pero valoran el servicio
- Contratos mes a mes pero leales

Hipótesis: Algunos clientes nuevos con precios altos son early adopters que valoran la calidad.

🎯 Recomendaciones de Negocio

1. Estrategias de Retención Proactiva

Para clientes de alto riesgo (probabilidad > 0.7):

- ✅ Contacto inmediato del equipo de retención
- ✅ Ofrecer descuentos personalizados (10-20%)
- ✅ Upgrade gratuito a servicios premium por 3 meses

Para clientes de riesgo moderado (probabilidad 0.4-0.7):

- ✅ Email marketing con ofertas de servicios adicionales
- ✅ Encuestas de satisfacción
- ✅ Incentivos para upgrade de contrato

---

2. Mejoras de Producto/Servicio

1. **Reducir precios de Fibra Óptica** o agregar más valor
    - Fibra óptica tiene alto churn a pesar de ser premium
2. **Bundling de servicios de seguridad**
    - Incluir OnlineSecurity y TechSupport en planes básicos
3. **Programa de lealtad para clientes nuevos**
    - Primeros 12 meses son críticos
4. **Incentivos para contratos largos**
    - Descuentos significativos por contratos de 1-2 años

---

3. Monitoreo Continuo

- **Dashboard en tiempo real** con scores de churn
- **Alertas automáticas** para clientes que cruzan umbral de riesgo
- **Re-entrenamiento mensual** del modelo con datos nuevos
- **A/B testing** de estrategias de retención

🔗 Relación con el Análisis General

Este bloque **cierra el ciclo completo**:

1. ✅ Problema definido (Introducción)
2. ✅ Datos explorados (EDA)
3. ✅ Features creadas (Feature Engineering)
4. ✅ Modelos entrenados (Baseline)
5. ✅ Desbalanceo manejado (SMOTE)
6. ✅ Hiperparámetros optimizados (GridSearch)
7. ✅ **Modelo evaluado y desplegado** (Este bloque)

💡 Puntos Clave para Recordar

1. **Modelo final**: ~83% accuracy, ~82% recall, ~0.87 AUC
2. **ROI positivo**: ~$205,000 de beneficio neto estimado
3. **Variables clave**: tenure, MonthlyCharges, Contract
4. **Falsos Negativos**: 68 clientes (costo: $102,000)
5. **Falsos Positivos**: 80 clientes (costo: $4,000)
6. **Recomendación**: Implementar sistema de alertas proactivo

---

11. Guardado del Modelo y Preparación para Deployment

🎯 Objetivo

Una vez que tenemos el modelo optimizado, necesitamos **guardarlo** para poder usarlo en producción sin tener que re-entrenarlo cada vez.

💾 Guardado del Modelo

El notebook utiliza **joblib** para serializar el modelo:

```
import joblib
from datetime import datetime

# Guardar el modelo optimizado
model_filename = 'best_rf_model.pkl'
joblib.dump(best_rf, model_filename)

# Guardar también el scaler y el encoder si se usaron
joblib.dump(scaler, 'scaler.pkl')
```

¿Por qué joblib y no pickle?

- **Más eficiente** para objetos grandes (como modelos de ML)
- **Mejor compresión** de arrays de NumPy
- **Más rápido** para serializar/deserializar

📊 Metadata del Modelo

El notebook también guarda metadata importante:

```
model_metadata = {
    'model_type': 'RandomForestClassifier',
    'metrics': {
        'roc_auc': 0.87,
        'recall': 0.83,
        'precision': 0.72,
        'f1_score': 0.77
    },
    'training_date': datetime.now().isoformat(),
    'features': list(X.columns),
    'n_features': len(X.columns),
    'n_samples_train': len(X_train),
    'hyperparameters': best_rf.get_params()
}

# Guardar metadata
import json
with open('model_metadata.json', 'w') as f:
    json.dump(model_metadata, f, indent=2)
```

📏 Tamaño del Modelo

El notebook reporta:

- **Tamaño del archivo**: ~50-100 MB (dependiendo de la configuración)
- **Estimación de RAM en producción**: ~200-300 MB

🚀 Consideraciones para Deployment

Opciones de Deployment:

1. **API REST** (Flask/FastAPI)
    - Crear endpoint para predicciones en tiempo real
    - Ejemplo: POST /predict con datos del cliente
2. **Batch Processing**

    - Procesar lotes de clientes periódicamente
    - Guardar scores en base de datos
3. **Cloud Deployment**
    - AWS SageMaker
    - Google Cloud AI Platform
    - Azure ML
    - Render/Railway (para proyectos pequeños)

Requerimientos Mínimos:

- **RAM**: 512 MB - 1 GB
- **CPU**: 1-2 cores
- **Almacenamiento**: 500 MB
- **Python**: 3.8+
- **Dependencias**: scikit-learn, pandas, numpy, joblib

🔄 Versionado del Modelo

Buenas prácticas:

```
# Incluir versión en el nombre del archivo
model_version = "v1.0.0"
model_filename = f'churn_model_{model_version}.pkl'
joblib.dump(best_rf, model_filename)
```

📝 Ejemplo de Uso en Producción

```
# Cargar el modelo
import joblib
model = joblib.load('best_rf_model.pkl')

# Hacer predicción para un nuevo cliente
new_customer = {
    'tenure': 12,
    'MonthlyCharges': 70.5,
    'Contract': 'Month-to-month',
    # ... otras features
}

# Preprocesar (aplicar mismo encoding que en entrenamiento)
new_customer_processed = preprocess(new_customer)

# Predecir
churn_probability = model.predict_proba(new_customer_processed)[0][1]
churn_prediction = model.predict(new_customer_processed)[0]

print(f"Probabilidad de churn: {churn_probability:.2%}")
print(f"Predicción: {'Churn' if churn_prediction == 1 else 'No Churn'}")
```

🎓 Conclusión Final del Proyecto

Hemos construido un sistema de predicción de churn que:

- ✅ Detecta 82% de clientes en riesgo
- ✅ Genera ROI positivo significativo

- ✅ Proporciona insights accionables
- ✅ Está listo para producción
- ✅ Está guardado y versionado correctamente

El valor real no está solo en el modelo, sino en las **acciones que permite tomar**: retener clientes proactivamente, optimizar precios, mejorar servicios y aumentar la rentabilidad del negocio.

Próximos pasos sugeridos:

1. Desplegar modelo en producción (API REST o batch)
2. Integrar con CRM para alertas automáticas
3. Implementar estrategias de retención
4. Monitorear resultados y re-entrenar periódicamente
5. Expandir análisis a segmentos específicos de clientes

---

12. Conclusiones Finales y Recomendaciones

🎯 Resumen del Proyecto Completo

A lo largo de este análisis exhaustivo, hemos completado un ciclo completo de ciencia de datos para resolver un problema crítico de negocio: la predicción y prevención del churn de clientes en telecomunicaciones.

📊 Logros Principales

1. Modelo Predictivo Robusto

- **Accuracy**: 83% - El modelo acierta en 8 de cada 10 predicciones
- **Recall**: 82% - Detecta 82% de los clientes que realmente harán churn
- **Precision**: 70% - Cuando predice churn, acierta en 7 de cada 10 casos
- **ROC-AUC**: 0.87 - Excelente capacidad de discriminación
- **F1-Score**: 76% - Balance óptimo entre precision y recall

2. Valor de Negocio Demostrado

ROI Estimado: ~$205,000 de beneficio neto

Desglose:

- Inversión en retención: $19,550
- Ahorro por clientes retenidos: $327,000
- Costo de falsos negativos: $102,000
- Costo de falsos positivos: $4,000

3. Insights Accionables

Factores de Riesgo Identificados:

1. **Contratos mes a mes**: 42% de churn (vs. 3% en contratos de 2 años)
2. **Clientes nuevos**: Tenure < 12 meses = alto riesgo
3. **Precios altos**: MonthlyCharges > $70 aumenta significativamente el riesgo
4. **Servicios limitados**: Sin OnlineSecurity, TechSupport = mayor churn
5. **Fibra óptica**: Paradójicamente, el servicio premium tiene más churn

🚀 Recomendaciones Estratégicas

A. Estrategias de Retención Inmediata

Para Clientes de Alto Riesgo (Score > 0.7):

1. Contacto proactivo del equipo de retención dentro de 24 horas
2. Descuentos personalizados del 15-20% por 6 meses
3. Upgrade gratuito a servicios premium por 3 meses
4. Asignación de account manager dedicado

Para Clientes de Riesgo Moderado (Score 0.4-0.7):

1. Campañas de email marketing con ofertas especiales
2. Encuestas de satisfacción para identificar puntos de dolor
3. Incentivos para agregar servicios adicionales
4. Programa de puntos de lealtad

B. Mejoras de Producto y Servicio

1. **Reestructurar Precios de Fibra Óptica**
    - Reducir precio o agregar más valor incluido
    - Bundle con servicios de seguridad sin costo adicional
    - Garantía de satisfacción de 90 días
2. **Programa de Onboarding para Nuevos Clientes**
    - Primeros 12 meses son críticos
    - Descuentos progresivos: 20% mes 1-3, 15% mes 4-6, 10% mes 7-12
    - Check-ins mensuales de satisfacción
    - Tutorial personalizado de servicios
3. **Bundling Inteligente**
    - Incluir OnlineSecurity y TechSupport en todos los planes
    - Paquetes familiares con descuentos significativos
    - Servicios de streaming incluidos en planes premium
4. **Incentivos para Contratos Largos**
    - 25% descuento en contratos de 2 años
    - 15% descuento en contratos de 1 año
    - Penalización mínima por cancelación anticipada

C. Implementación Técnica

1. **Dashboard en Tiempo Real**
    - Scores de churn actualizados diariamente
    - Alertas automáticas para clientes que cruzan umbrales
    - Segmentación por nivel de riesgo
    - KPIs de retención por equipo
2. **Integración con CRM**
    - API para scoring en tiempo real
    - Historial de interacciones con clientes de riesgo
    - Tracking de efectividad de estrategias de retención
    - Automatización de campañas según score
3. **Re-entrenamiento del Modelo**
    - Actualización mensual con datos nuevos
    - Monitoreo de drift del modelo
    - A/B testing de nuevas features

    - Validación continua de performance
4. **Expansión del Análisis**
    - Segmentación por tipo de cliente (residencial, empresarial)
    - Análisis de lifetime value (LTV)
    - Predicción de upsell/cross-sell
    - Análisis de sentimiento de interacciones

📈 Métricas de Éxito

KPIs para Monitorear:

1. **Tasa de Churn General**
    - Objetivo: Reducir de 27% a 20% en 12 meses
    - Métrica: % de clientes que cancelan mensualmente
2. **Efectividad de Retención**
    - Objetivo: Retener 70% de clientes contactados
    - Métrica: % de clientes de alto riesgo que NO hacen churn después de intervención
3. **ROI de Campañas**
    - Objetivo: Mantener ROI > 300%
    - Métrica: (Valor retenido - Costo de retención) / Costo de retención
4. **Precisión del Modelo**
    - Objetivo: Mantener Recall > 80%
    - Métrica: Monitoreo mensual de métricas del modelo

🎓 Lecciones Aprendidas

1. **El Desbalanceo de Clases es Crítico**
    - SMOTE mejoró el Recall en ~30%
    - Sin balanceo, el modelo ignoraba la clase minoritaria
2. **Feature Engineering Marca la Diferencia**
    - Variables derivadas (TenureGroup, IsPremium) mejoraron significativamente el modelo
    - El conocimiento del dominio es tan importante como los algoritmos
3. **La Métrica Correcta es Fundamental**
    - Accuracy puede ser engañosa con datos desbalanceados
    - Para churn, Recall es más importante que Precision
4. **Ensemble Methods Dominan**
    - XGBoost y Random Forest superaron consistentemente a modelos simples
    - La optimización de hiperparámetros dio mejoras modestas pero valiosas (3-5%)
5. **El Valor está en la Acción**
    - Un modelo perfecto sin implementación no vale nada
    - Las recomendaciones accionables son tan importantes como las predicciones

🔮 Próximos Pasos

Corto Plazo (1-3 meses):

1. ✅ Desplegar modelo en producción

2. ✅ Implementar dashboard de monitoreo
3. ✅ Lanzar programa piloto de retención
4. ✅ Capacitar equipos de ventas y retención

Mediano Plazo (3-6 meses):

1. ✅ Evaluar resultados del programa piloto
2. ✅ Ajustar estrategias según feedback
3. ✅ Expandir a todos los segmentos de clientes
4. ✅ Implementar re-entrenamiento automático

Largo Plazo (6-12 meses):

1. ✅ Desarrollar modelos específicos por segmento
2. ✅ Integrar análisis de sentimiento
3. ✅ Predicción de LTV y propensión a upsell
4. ✅ Optimización continua basada en resultados

🌟 Reflexión Final

Este proyecto demuestra el poder del Machine Learning aplicado a problemas reales de negocio. No se trata solo de construir un modelo preciso, sino de:

- **Entender el problema** profundamente
- **Explorar los datos** exhaustivamente
- **Crear features** inteligentes
- **Seleccionar métricas** apropiadas
- **Generar insights** accionables
- **Implementar soluciones** prácticas

El verdadero éxito se medirá no en la precisión del modelo, sino en cuántos clientes logramos retener y cuánto valor generamos para el negocio.

📊 Resumen Técnico del Proyecto

🛠️ Tecnologías Utilizadas

Lenguaje y Entorno: - Python 3.8+ - Jupyter Notebook / Google Colab

Librerías de Análisis de Datos: - **Pandas** (1.3+): Manipulación y análisis de datos - **NumPy** (1.21+): Operaciones numéricas y arrays - **Matplotlib** (3.4+): Visualizaciones básicas - **Seaborn** (0.11+): Visualizaciones estadísticas avanzadas

Librerías de Machine Learning: - **Scikit-learn** (1.0+): Algoritmos de ML, preprocesamiento, métricas - **XGBoost** (1.5+): Gradient Boosting optimizado - **imbalanced-learn** (0.8+): SMOTE para balanceo de clases

Otras Herramientas: - **joblib**: Serialización de modelos - **warnings**: Supresión de advertencias - **datetime**: Manejo de fechas y timestamps

🔬 Metodologías Aplicadas

1. Análisis Exploratorio de Datos (EDA): - Análisis univariado de variables categóricas y numéricas - Análisis bivariado (relación features vs target) - Visualizaciones: histogramas, boxplots, heatmaps, barplots - Detección de outliers y valores faltantes - Análisis de correlaciones

2. Pruebas de Hipótesis Estadísticas: - **Chi-cuadrado**: Variables categóricas vs Churn - **Mann-Whitney U**: Variables numéricas vs Churn - Nivel de significancia: α = 0.05 - 7 hipótesis probadas y validadas

3. Feature Engineering: - Creación de variables derivadas: - TenureGroup: Categorización de tenure - IsPremium: Indicador de servicios premium - HasMultipleServices: Número de servicios contratados - Encoding de variables categóricas (Label Encoding, One-Hot Encoding) - Escalado de variables numéricas (StandardScaler)

4. Manejo de Desbalanceo de Clases: - **SMOTE** (Synthetic Minority Over-sampling Technique) - Balanceo de clases: 73% No Churn → 50% No Churn - Mejora significativa en Recall: 50% → 78%

5. Modelado y Evaluación: - **Algoritmos probados**: - Logistic Regression (baseline) - Decision Tree - Random Forest ⭐ (mejor modelo) - Gradient Boosting - XGBoost - SVM - KNN

- **Optimización de Hiperparámetros**:
  - GridSearchCV con validación cruzada (5-fold)
  - Espacio de búsqueda exhaustivo
  - Métrica de optimización: ROC-AUC
- **Validación**:
  - Train-Test Split (80-20)
  - Stratified K-Fold Cross-Validation
  - Métricas múltiples: Accuracy, Precision, Recall, F1, ROC-AUC

📈 Métricas de Evaluación

Métricas Principales: - **ROC-AUC**: 0.87 - Capacidad de discriminación - **Recall**: 83% - Detección de churners - **Precision**: 72% - Precisión de predicciones positivas - **F1-Score**: 77% - Balance precision-recall - **Accuracy**: 83% - Precisión general

Interpretación de Métricas: - **Recall alto** (83%): Detectamos la mayoría de clientes en riesgo - **Precision aceptable** (72%): Minimizamos falsos positivos - **ROC-AUC excelente** (0.87): Modelo discrimina muy bien entre clases

🎯 Resultados de Negocio

ROI Estimado: ~$205,000 de beneficio neto

Desglose Financiero: - Inversión en retención: $19,550 - Ahorro por clientes retenidos: $327,000 - Costo de falsos negativos: $102,000 - Costo de falsos positivos: $4,000

Impacto Esperado: - Reducción de churn: 27% → 20% (objetivo 12 meses) - Tasa de retención: 70% de clientes contactados - ROI de campañas: > 300%

📚 Referencias y Recursos Adicionales

Librerías Utilizadas:

- Pandas: https://pandas.pydata.org/
- NumPy: https://numpy.org/

- Scikit-learn: https://scikit-learn.org/
- XGBoost: https://xgboost.readthedocs.io/
- Seaborn: https://seaborn.pydata.org/
- Matplotlib: https://matplotlib.org/
- imbalanced-learn: https://imbalanced-learn.org/

Conceptos Clave:

- SMOTE: https://arxiv.org/abs/1106.1813
- Random Forest: https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf
- XGBoost: https://arxiv.org/abs/1603.02754
- Cross-Validation: https://scikit-learn.org/stable/modules/cross_validation.html

---

📝 Notas Finales

Autor: Análisis de Customer Churn - Proyecto de Ciencia de Datos **Fecha de Creación**: 2025-11-20 **Última Actualización**: 2025-11-21 **Versión**: 1.1 **Dataset**: Telco Customer Churn (7,043 registros, 21 variables)

Cambios en v1.1: - ✅ Agregada sección detallada de Guardado del Modelo y Deployment - ✅ Agregado Resumen Técnico completo del proyecto - ✅ Actualizadas métricas finales del modelo (Recall: 83%, Precision: 72%) - ✅ Sincronizado con el notebook Telco_Customer_Churn.ipynb

Contacto: Para preguntas o consultas sobre este análisis, por favor contactar al equipo de Data Science.

FIN DEL DOCUMENTO