

Preguntas de Sustentación - Proyecto Telco Customer Churn

Índice de Categorías

1. [Comprendión del Problema de Negocio](#)
 2. [Exploración y Análisis de Datos](#)
 3. [Preprocesamiento y Limpieza de Datos](#)
 4. [Feature Engineering](#)
 5. [Modelado y Entrenamiento](#)
 6. [Evaluación y Métricas](#)
 7. [Conclusiones y Recomendaciones](#)
-

1. Comprendión del Problema de Negocio

Pregunta 1: ¿Qué es el “Customer Churn” y por qué es importante para una empresa de telecomunicaciones?

Respuesta:

El “Customer Churn” (o abandono de clientes) es cuando un cliente decide dejar de usar los servicios de una empresa y se va con la competencia o simplemente cancela el servicio.

Analogía cotidiana: Imagina que tienes una cafetería. El churn sería como los clientes que solían venir todos los días pero de repente dejan de hacerlo y empiezan a ir a la cafetería de enfrente.

Para una empresa de telecomunicaciones es crítico porque:

- **Cuesta más caro conseguir un cliente nuevo que mantener uno existente** (como es más fácil que un amigo vuelva a tu cafetería que convencer a un desconocido de que la visite)
- **Los clientes que se van representan pérdida de ingresos mensuales recurrentes**
- **Pueden irse a la competencia**, fortaleciendo a otros
- **Identificar clientes en riesgo permite tomar acciones preventivas** (como ofrecer descuentos o mejores planes)

En nuestro proyecto, el 27% de los clientes abandonaron el servicio, lo que representa una pérdida significativa de ingresos.

Pregunta 2: ¿Cuál es el objetivo principal de este proyecto de Machine Learning?

Respuesta:

El objetivo principal es **desarrollar un modelo predictivo que identifique con anticipación qué clientes tienen alta probabilidad de abandonar el servicio**, para que la empresa pueda tomar acciones preventivas de retención.

Analogía cotidiana: Es como tener un “detector de problemas” en tu cafetería que te avisa: “Este cliente que viene todos los días está empezando a venir menos, probablemente está pensando en ir a otro lugar”. Con esa información, podrías ofrecerle un café gratis o preguntarle si algo no le gusta.

Específicamente, nuestro modelo busca:

1. **Predecir** qué clientes se irán antes de que lo hagan
2. **Identificar** los factores que más influyen en la decisión de irse
3. **Permitir** que la empresa diseñe estrategias de retención personalizadas
4. **Optimizar recursos** enfocándose en los clientes con mayor riesgo

El modelo no evita el churn directamente, pero da a la empresa la información necesaria para actuar a tiempo.

Pregunta 3: ¿Qué información contiene el dataset y cuántos clientes incluye?

Respuesta:

El dataset contiene información de **7,043 clientes** de una empresa de telecomunicaciones, con **21 variables** (columnas) que describen diferentes aspectos de cada cliente.

Tipos de información incluida:

1. **Información demográfica:**

- o Género (Male/Female)

- o Si es adulto mayor (SeniorCitizen)
- o Si tiene pareja (Partner)
- o Si tiene dependientes (Dependents)

2. Información de servicios contratados:

- o Servicio telefónico
- o Tipo de internet (DSL, Fiber optic, o ninguno)
- o Servicios adicionales (streaming, seguridad online, soporte técnico, etc.)

3. Información de cuenta:

- o Tiempo como cliente (tenure - en meses)
- o Tipo de contrato (mes a mes, 1 año, 2 años)
- o Método de pago
- o Cargos mensuales y totales

4. Variable objetivo:

- o Churn (Yes/No) - si el cliente abandonó o no el servicio

Analogía: Es como tener una ficha completa de cada cliente de tu cafetería: edad, si viene solo o con familia, qué consume, cuánto tiempo lleva viniendo, cómo paga, y si sigue siendo cliente o dejó de venir.

Pregunta 4: ¿Qué significa que el dataset está “desbalanceado” y cómo afecta esto al proyecto?

Respuesta:

Un dataset desbalanceado significa que **no hay la misma cantidad de ejemplos de cada clase**. En nuestro caso:

- **73% de clientes NO abandonaron** (clase mayoritaria - “No”)
- **27% de clientes SÍ abandonaron** (clase minoritaria - “Yes”)

Analogía cotidiana: Imagina que quieres entrenar a alguien para identificar billetes falsos, pero le muestras 73 billetes verdaderos y solo 27 falsos. La persona aprenderá mucho mejor a reconocer billetes verdaderos que falsos, simplemente porque vio muchos más ejemplos de verdaderos.

Cómo afecta al proyecto:

1. **El modelo tiende a predecir la clase mayoritaria:** Si el modelo simplemente dijera “nadie se va” todo el tiempo, tendría 73% de precisión, pero sería inútil.
 2. **Dificultad para detectar churn:** Como hay menos ejemplos de clientes que se van, el modelo tiene menos información para aprender a identificarlos.
 3. **Necesidad de técnicas especiales:** Por eso usamos SMOTE (Synthetic Minority Over-sampling Technique) que crea ejemplos sintéticos de la clase minoritaria para balancear el dataset.
 4. **Métricas especiales:** No podemos usar solo “accuracy” (precisión general), necesitamos métricas como Recall y ROC-AUC que evalúen mejor la detección de la clase minoritaria.
-

2. Exploración y Análisis de Datos

Pregunta 5: ¿Qué es el Análisis Exploratorio de Datos (EDA) y por qué es importante?

Respuesta:

El Análisis Exploratorio de Datos (EDA) es como **hacer una investigación detallada antes de tomar decisiones importantes**. Es el proceso de examinar, visualizar y entender los datos antes de construir modelos.

Analogía cotidiana: Es como cuando vas a comprar un auto usado. Antes de decidir, lo inspeccionas: miras el motor, pruebas los frenos, revisas el kilometraje, verificas si tiene abolladuras. No comprarías el auto sin hacer esta inspección, ¿verdad? Lo mismo pasa con los datos.

En nuestro proyecto, el EDA nos permitió:

1. **Entender la distribución de churn:** Descubrimos que 27% de clientes se van (desbalanceo)
2. **Identificar patrones importantes:**
 - o Clientes con contratos mes a mes tienen mayor churn
 - o Clientes nuevos (tenure bajo) se van más
 - o El tipo de servicio de internet influye en el churn

3. Detectar problemas en los datos:

- o Encontramos 11 registros con TotalCharges vacío
- o Descubrimos que TotalCharges estaba como texto en lugar de número

4. Descubrir relaciones entre variables:

- o Correlación entre MonthlyCharges y TotalCharges
- o Relación entre servicios adicionales y retención

5. Guiar decisiones posteriores:

- o Qué variables son más importantes
- o Qué transformaciones necesitamos hacer
- o Qué modelos podrían funcionar mejor

Sin EDA, estaríamos construyendo modelos “a ciegas”, sin entender realmente nuestros datos.

Pregunta 6: ¿Qué visualizaciones utilizaste en el EDA y qué información te proporcionaron?

Respuesta:

Utilizamos varias visualizaciones, cada una con un propósito específico:

1. Gráficos de Barras (para variables categóricas):

- **Qué muestran:** Distribución de categorías y su relación con churn
- **Ejemplo en el proyecto:** Comparamos churn entre tipos de contrato (mes a mes vs. 1 año vs. 2 años)
- **Descubrimiento:** Los contratos mes a mes tienen muchísimo más churn
- **Analogía:** Como comparar cuántos clientes de tu cafetería pagan con efectivo vs. tarjeta

2. Histogramas (para variables numéricas):

- **Qué muestran:** Cómo se distribuyen los valores numéricos
- **Ejemplo:** Distribución de tenure (tiempo como cliente)
- **Descubrimiento:** Muchos clientes nuevos (0-12 meses) y muchos clientes antiguos (60+ meses)

- **Analogía:** Como ver cuántos clientes de tu cafetería llevan 1 mes, 6 meses, 1 año, etc.

3. Boxplots (diagramas de caja):

- **Qué muestran:** Valores mínimos, máximos, medianas y valores atípicos
- **Ejemplo:** MonthlyCharges comparando clientes que se van vs. los que se quedan
- **Descubrimiento:** Clientes con cargos muy altos o muy bajos tienen más churn
- **Analogía:** Como ver el rango de precios que pagan tus clientes y detectar extremos

4. Matriz de Correlación (heatmap):

- **Qué muestra:** Relaciones numéricas entre variables
- **Ejemplo:** Relación entre tenure, MonthlyCharges y TotalCharges
- **Descubrimiento:** TotalCharges está muy correlacionado con tenure (lógico: más tiempo = más pago total)
- **Analogía:** Como descubrir que los clientes que compran café también suelen comprar pastel

5. Curvas ROC y Precision-Recall:

- **Qué muestran:** Rendimiento del modelo en diferentes umbrales
 - **Uso:** Evaluar qué tan bien el modelo distingue entre clientes que se van y los que se quedan
-

Pregunta 7: ¿Qué es la correlación y qué descubriste al analizar las correlaciones en este proyecto?

Respuesta:

La correlación mide **qué tan relacionadas están dos variables numéricas**. Va de -1 a +1:

- **+1:** Correlación positiva perfecta (cuando una sube, la otra también)
- **0:** No hay correlación (son independientes)
- **-1:** Correlación negativa perfecta (cuando una sube, la otra baja)

Analogía cotidiana:

- **Correlación positiva:** Horas de estudio y calificaciones (más estudio → mejores notas)
- **Correlación negativa:** Horas de sueño y ojeras (más sueño → menos ojeras)
- **Sin correlación:** Talla de zapatos y habilidad en matemáticas

Descubrimientos en nuestro proyecto:

1. **TotalCharges y tenure (correlación alta positiva ~0.83):**
 - o Tiene sentido: más meses como cliente = más pago acumulado
 - o Es una relación casi matemática
2. **MonthlyCharges y TotalCharges (correlación moderada):**
 - o Clientes que pagan más al mes tienden a tener totales más altos
 - o Pero no es perfecta porque depende también del tiempo
3. **Variables categóricas codificadas:**
 - o Algunas mostraron correlación con churn
 - o Por ejemplo, tipo de contrato correlaciona con probabilidad de irse

Importancia para el proyecto:

- **Evitar redundancia:** Si dos variables están muy correlacionadas, quizás solo necesitamos una
 - **Identificar predictores:** Variables correlacionadas con churn son candidatas importantes
 - **Entender relaciones:** Nos ayuda a comprender el comportamiento del negocio
-

Pregunta 8: ¿Qué patrones importantes identificaste sobre los clientes que abandonan el servicio?

Respuesta:

Identificamos varios patrones claros que caracterizan a los clientes con mayor riesgo de churn:

1. Tiempo como cliente (Tenure):

- **Patrón:** Clientes nuevos (0-12 meses) tienen MUCHO más churn

- **Analogía:** Como en un gimnasio, la mayoría de personas que se dan de baja lo hacen en los primeros meses
- **Implicación:** Necesitamos estrategias especiales para retener clientes nuevos

2. Tipo de Contrato:

- **Patrón:** Contratos mes a mes tienen churn altísimo vs. contratos de 1-2 años
- **Analogía:** Es como Netflix vs. comprar una membresía anual de gimnasio. Con Netflix puedes cancelar cuando quieras, con el gimnasio estás comprometido
- **Implicación:** Incentivar contratos largos reduce churn

3. Servicios Adicionales:

- **Patrón:** Clientes SIN servicios como TechSupport u OnlineSecurity se van más
- **Analogía:** Como en un banco, si solo tienes cuenta de ahorros es más fácil irte que si tienes cuenta, tarjeta de crédito, inversiones, etc.
- **Implicación:** Más servicios = más “enganchado” está el cliente

4. Cargos Mensuales:

- **Patrón:** Clientes con MonthlyCharges muy altos tienen más churn
- **Analogía:** Si pagas mucho por algo, eres más crítico y exigente. Si no ves el valor, te vas
- **Implicación:** Revisar estrategia de precios para clientes de alto valor

5. Tipo de Internet:

- **Patrón:** Clientes con Fiber Optic tienen más churn que DSL
- **Possible razón:** Fiber es más caro, expectativas más altas
- **Implicación:** Mejorar experiencia y valor percibido del servicio premium

6. Método de Pago:

- **Patrón:** Electronic check tiene más churn que pagos automáticos
 - **Razón:** Pagos automáticos crean “fricción” para cancelar
 - **Implicación:** Incentivar pagos automáticos
-

Pregunta 9: ¿Cómo interpretaste las estadísticas descriptivas del dataset?

Respuesta:

Las estadísticas descriptivas nos dan un “resumen numérico” de cada variable. Veamos las más importantes:

Tenure (tiempo como cliente en meses):

Media: 32 meses

Mediana: 29 meses

Mínimo: 0 meses

Máximo: 72 meses

Interpretación:

- El cliente promedio lleva casi 3 años
- Hay clientes muy nuevos (0 meses) y muy antiguos (6 años)
- La distribución es bastante amplia

Analogía: Como medir cuánto tiempo llevan tus amigos en Facebook. Algunos acaban de unirse, otros están desde 2008.

MonthlyCharges (cargo mensual):

Media: \$64.76

Mínimo: \$18.25

Máximo: \$118.75

Interpretación:

- Gran variabilidad en lo que pagan los clientes
- Hay planes económicos y planes premium
- El promedio está más cerca del máximo (muchos clientes en planes caros)

SeniorCitizen (adultos mayores):

Media: 0.16 (16%)

Interpretación:

- Solo 16% son adultos mayores
- La mayoría de clientes son menores de 65 años
- Dataset desbalanceado también en esta variable

TotalCharges (cargo total acumulado):

- Varía mucho porque depende de tenure y MonthlyCharges
- Clientes nuevos tienen totales bajos, antiguos tienen totales altos

¿Por qué es importante?

1. **Detectar valores extraños:** Si viéramos tenure de 500 meses, sabríamos que hay un error
 2. **Entender rangos:** Nos dice qué valores son normales y cuáles son atípicos
 3. **Planear transformaciones:** Si una variable tiene valores muy dispersos, quizás necesitamos normalizarla
 4. **Contexto de negocio:** Entendemos mejor el perfil típico del cliente
-

3. Preprocesamiento y Limpieza de Datos

Pregunta 10: ¿Qué problema encontraste con la variable TotalCharges y cómo lo resolviste?

Respuesta:

Encontramos **dos problemas** con la variable TotalCharges:

Problema 1: Tipo de dato incorrecto

- TotalCharges estaba guardado como “texto” (object) en lugar de número (float)
- Esto impide hacer cálculos matemáticos

Analogía: Es como si en una hoja de Excel tuvieras precios escritos como texto (“\$100”) en lugar de números (100). No podrías sumarlos.

Problema 2: Valores en blanco

- 11 registros tenían TotalCharges como espacio en blanco (“ ”)
- Estos eran clientes con tenure = 0 (clientes nuevísimos)

Solución aplicada:

```
# Paso 1: Convertir espacios en blanco a NaN (valor faltante)
df['TotalCharges'] = df['TotalCharges'].replace(' ', np.nan)
```

```
# Paso 2: Convertir a numérico
```

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

```
# Paso 3: Imputar valores faltantes  
# Para clientes nuevos (tenure=0), TotalCharges = MonthlyCharges  
df.loc[df['TotalCharges'].isna(), 'TotalCharges'] = \  
    df.loc[df['TotalCharges'].isna(), 'MonthlyCharges']
```

Razonamiento de la solución:

- Si un cliente acaba de contratar (tenure=0), su cargo total debe ser igual a su cargo mensual
- Es lógico y coherente con el negocio
- No inventamos datos, usamos información que ya teníamos

Alternativas que NO usamos:

- X Eliminar los 11 registros (perderíamos información valiosa)
- X Poner ceros (sería incorrecto, sí tienen un cargo)
- X Poner la media (no tendría sentido para clientes nuevos)

Verificación:

Después de la imputación, confirmamos que no quedaron valores faltantes (0 NaN).

Pregunta 11: ¿Qué es la codificación de variables categóricas y por qué es necesaria?

Respuesta:

La codificación de variables categóricas es **convertir texto en números** para que los algoritmos de Machine Learning puedan procesarlos.

¿Por qué es necesaria? Los algoritmos de ML trabajan con matemáticas (sumas, multiplicaciones, etc.). No pueden hacer matemáticas con palabras como “Male”, “Female”, “DSL”, etc.

Analogía cotidiana: Es como traducir un libro del español al inglés para que alguien que solo habla inglés pueda leerlo. El contenido es el mismo, solo cambia el “idioma”.

Tipos de codificación que usamos:

1. Label Encoding (para variables binarias):

- Convierte dos categorías en 0 y 1
- Ejemplo: “Male” → 0, “Female” → 1
- Usado para: gender, Partner, Dependents, PhoneService, PaperlessBilling, Churn

Analogía: Como un interruptor de luz: apagado=0, encendido=1

2. One-Hot Encoding (para variables con múltiples categorías):

- Crea una columna nueva por cada categoría
- Ejemplo: InternetService tiene 3 valores (DSL, Fiber optic, No)
 - Se convierte en 3 columnas: InternetService_DSL, InternetService_Fiber, InternetService_No
 - Si un cliente tiene DSL: [1, 0, 0]
 - Si tiene Fiber: [0, 1, 0]
 - Si no tiene: [0, 0, 1]

Analogía: Como marcar casillas en un formulario. Si te preguntan “¿Qué mascotas tienes?” y las opciones son Perro, Gato, Pájaro, marcas las que aplican.

Variables que usaron One-Hot Encoding:

- InternetService (3 categorías)
- Contract (3 categorías)
- PaymentMethod (4 categorías)
- MultipleLines, OnlineSecurity, OnlineBackup, etc.

¿Por qué no usar Label Encoding para todo?

Si codificáramos Contract como: Month-to-month=0, One year=1, Two year=2, el modelo podría pensar que “Two year” es el doble de “One year” matemáticamente, lo cual no tiene sentido. One-Hot Encoding evita este problema.

Pregunta 12: ¿Qué es la normalización/estandarización y por qué la aplicaste?

Respuesta:

La normalización (o estandarización) es **poner todas las variables numéricas en la misma escala** para que ninguna domine sobre las otras.

El problema sin normalización:

Imagina estas dos variables:

- **Tenure:** rango de 0 a 72 meses
- **MonthlyCharges:** rango de 18 a 118 dólares
- **TotalCharges:** rango de 18 a 8,684 dólares

TotalCharges tiene números MUCHO más grandes. Algunos algoritmos (como KNN o SVM) podrían pensar que TotalCharges es más importante solo porque sus números son más grandes, aunque no sea cierto.

Analogía cotidiana:

Imagina que evalúas restaurantes con dos criterios: - Calidad de comida (escala 1-5 estrellas) - Precio (escala \$5-\$100)

Si simplemente sumas, el precio dominará porque sus números son más grandes. Necesitas ponerlos en la misma escala primero.

Solución: StandardScaler

Usamos StandardScaler de scikit-learn que transforma cada variable para que tenga:

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

Fórmula: $\text{valor_normalizado} = (\text{valor} - \text{media}) / \text{desviación_estándar}$

Ejemplo práctico:

Tenure original: [1, 34, 2, 45, 2, ...]

Tenure normalizado: [-1.28, 0.07, -1.24, 0.51, -1.24, ...]

¿Qué modelos necesitan normalización?

- **✓ SÍ necesitan:** KNN, SVM, Regresión Logística, Redes Neuronales
- **✗ NO necesitan:** Random Forest, Decision Trees, XGBoost (son invariantes a escala)

En nuestro proyecto:

Aplicamos StandardScaler a las variables numéricas (tenure, MonthlyCharges, TotalCharges) dentro de nuestro pipeline de preprocesamiento, asegurando que se aplique correctamente en entrenamiento y prueba.

Pregunta 13: ¿Qué es un pipeline de preprocesamiento y qué ventajas ofrece?

Respuesta:

Un pipeline es una “**cadena de montaje automatizada**” que aplica todos los pasos de preprocesamiento en el orden correcto.

Analogía cotidiana: Es como una receta de cocina automatizada. En lugar de hacer cada paso manualmente (picar, mezclar, hornear), tienes una máquina que hace todo en secuencia. Solo pones los ingredientes crudos y sale el plato terminado.

Nuestro pipeline hace:

```
Pipeline([
    ('preprocessor', ColumnTransformer([
        ('num', StandardScaler(), numerical_features),
        ('cat', OneHotEncoder(), categorical_features)
    ]),
    ('classifier', RandomForestClassifier())
])
```

Paso a paso:

1. **Separa** variables numéricas y categóricas
2. **Normaliza** las numéricas con StandardScaler
3. **Codifica** las categóricas con OneHotEncoder
4. **Combina** todo en un solo dataset procesado
5. **Entrena** el modelo con los datos procesados

Ventajas del pipeline:

1. Evita Data Leakage (fuga de datos):

- **Problema sin pipeline:** Si normalizas antes de dividir train/test, el test “ve” información del train

- **Solución con pipeline:** La normalización se aprende SOLO con datos de entrenamiento

Analogía: Es como estudiar para un examen. No puedes ver las respuestas del examen antes de tomarlo (eso sería trampa). El pipeline asegura que el test sea realmente “nuevo” para el modelo.

2. Reproducibilidad:

- Los mismos pasos se aplican siempre en el mismo orden
- Fácil de compartir y replicar

3. Código más limpio:

- Todo en un solo objeto
- Menos posibilidad de errores

4. Facilita predicciones futuras:

- Cuando llegue un cliente nuevo, el pipeline aplica automáticamente todos los pasos
- No tienes que recordar “primero normalizo, luego codifico, etc.”

5. Compatibilidad con validación cruzada:

- El pipeline se ajusta correctamente en cada fold
 - Evita errores comunes
-

Pregunta 14: ¿Por qué dividiste los datos en conjuntos de entrenamiento y prueba?

Respuesta:

Dividir los datos es como **separar las preguntas de estudio de las preguntas del examen final**.

La división:

- **80% Entrenamiento (train):** El modelo aprende de estos datos
- **20% Prueba (test):** El modelo es evaluado con estos datos (nunca los vio antes)

¿Por qué es necesario?

Analogía del estudiante:

Imagina que estudias para un examen con 100 preguntas. Si el examen final tiene EXACTAMENTE las mismas 100 preguntas que estudiaste, podrías sacar 100% solo memorizando, sin realmente entender. Pero si el examen tiene preguntas nuevas (pero del mismo tema), ahí se ve si realmente aprendiste.

En Machine Learning:

Sin división (ERROR):

Entrenar con TODO → Evaluar con TODO = 100% accuracy

Pero es trampa, el modelo solo memorizó. Cuando lleguen clientes nuevos reales, fallará.

Con división (CORRECTO):

Entrenar con 80% → Evaluar con 20% nuevo = 85% accuracy

Esta es la verdadera capacidad del modelo con datos que nunca vio.

Conceptos clave:

1. Overfitting (sobreajuste):

- El modelo memoriza los datos de entrenamiento
- Funciona perfecto en train, mal en test
- **Analogía:** Estudiante que memoriza respuestas pero no entiende

2. Generalización:

- El modelo aprende patrones generales
- Funciona bien tanto en train como en test
- **Analogía:** Estudiante que entiende los conceptos y puede resolver problemas nuevos

Detalles técnicos de nuestra división:

```
X_train, X_test, y_train, y_test = train_test_split(  
    X, y,  
    test_size=0.2,      # 20% para test  
    random_state=42,   # Reproducibilidad  
    stratify=y         # Mantiene proporción de churn  
)
```

stratify=y es importante porque:

- Asegura que train y test tengan la misma proporción de churn (27%)
 - Sin esto, podríamos tener 30% churn en train y 20% en test (desbalance)
-

4. Feature Engineering

Pregunta 15: ¿Qué es Feature Engineering y por qué es importante?

Respuesta:

Feature Engineering es el **arte de crear nuevas variables (características) a partir de las existentes** para ayudar al modelo a aprender mejor.

Analogía cotidiana:

Imagina que quieres predecir si un estudiante aprobará un examen. Tienes: - Horas de estudio por día - Días hasta el examen

Podrías crear una nueva variable: **Total de horas de estudio = horas/día × días**

Esta nueva variable podría ser más útil que las originales por separado.

En nuestro proyecto, creamos 6 nuevas características:

1. TenureGroup (Grupo de antigüedad):

0-12 meses → "0-1 year"

13-24 meses → "1-2 years"

25-48 meses → "2-4 years"

49-72 meses → "4-6 years"

Por qué: Es más fácil para el modelo entender “cliente nuevo” vs. “cliente antiguo” que números exactos.

Analogía: Como clasificar edades en “niño”, “adolescente”, “adulto” en lugar de usar edad exacta.

2. AvgMonthlyCharges (Cargo mensual promedio):

AvgMonthlyCharges = TotalCharges / (tenure + 1)

Por qué: Nos dice cuánto paga en promedio cada mes. Un cliente que lleva 10 meses y pagó \$1000 total, paga \$100/mes en promedio.

3. ChargeRatio (Ratio de cargos):

ChargeRatio = MonthlyCharges / (AvgMonthlyCharges + 0.01)

Por qué: Detecta si el cargo actual es más alto o más bajo que el promedio histórico. Si es >1, está pagando más ahora (quizás por eso se quiere ir).

4. TotalServices (Total de servicios contratados):

Suma de: PhoneService + InternetService + OnlineSecurity +
OnlineBackup + DeviceProtection + TechSupport +
StreamingTV + StreamingMovies

Por qué: Clientes con más servicios están más “enganchados” y tienen menos churn.

Analogía: Como en un banco, si solo tienes cuenta de ahorros es fácil irte, pero si tienes cuenta, tarjeta, crédito hipotecario, inversiones... es más difícil.

5. HasMultipleServices (Tiene múltiples servicios):

1 si TotalServices >= 3, sino 0

Por qué: Versión binaria de TotalServices, más simple para algunos modelos.

6. IsNewCustomer (Es cliente nuevo):

1 si tenure <= 12 meses, sino 0

Por qué: Sabemos que clientes nuevos tienen mucho más churn. Esta variable lo hace explícito.

Importancia del Feature Engineering:

- Puede mejorar el rendimiento del modelo más que cambiar de algoritmo
- Incorpora conocimiento del negocio al modelo
- Hace que patrones complejos sean más evidentes
- Es donde la creatividad y experiencia del científico de datos brillan

Pregunta 16: ¿Cómo decidiste qué nuevas características crear?

Respuesta:

La creación de características se basó en **tres fuentes principales**:

1. Análisis Exploratorio de Datos (EDA):

- Observamos que tenure era muy importante → creamos TenureGroup e IsNewCustomer
- Vimos que clientes con más servicios tenían menos churn → creamos TotalServices
- Notamos patrones en los cargos → creamos AvgMonthlyCharges y ChargeRatio

2. Conocimiento del Negocio:

- **Lógica de telecomunicaciones:** Sabemos que clientes nuevos son más volátiles
- **Comportamiento del consumidor:** Más servicios = más compromiso
- **Análisis financiero:** Cambios en el cargo mensual pueden indicar insatisfacción

Analogía: Es como un doctor que usa síntomas (datos) + conocimiento médico (negocio) para diagnosticar.

3. Intuición y Experimentación:

- Probamos diferentes combinaciones
- Evaluamos si mejoraban el modelo
- Descartamos las que no aportaban valor

Proceso de validación:

Para cada nueva característica nos preguntamos:

1. **¿Tiene sentido de negocio?** ✓
2. **¿Aporta información nueva?** (no es redundante) ✓
3. **¿Mejora el rendimiento del modelo?** ✓
4. **¿Es calculable para clientes futuros?** ✓

Ejemplo de característica que NO creamos:

- “Días hasta que se fue el cliente” → Solo sabemos esto DESPUÉS de que se va, no sirve para predecir

Características que SÍ creamos y por qué:

Característica	Razonamiento
TenureGroup	Simplifica patrones temporales
AvgMonthlyCharges	Normaliza por tiempo como

Característica	Razonamiento
	cliente
ChargeRatio	Detecta cambios en facturación
TotalServices	Mide nivel de compromiso
HasMultipleServices	Versión binaria más simple
IsNewCustomer	Hace explícito un patrón clave

Pregunta 17: ¿Podrías dar un ejemplo concreto de cómo una característica creada ayuda al modelo?

Respuesta:

Veamos el ejemplo de **TotalServices** en detalle:

Datos originales (fragmento):

Cliente A:

- PhoneService: Yes
- InternetService: Yes
- OnlineSecurity: No
- OnlineBackup: No
- TechSupport: No
- StreamingTV: No
- Churn: Yes (se fue)

Cliente B:

- PhoneService: Yes
- InternetService: Yes
- OnlineSecurity: Yes
- OnlineBackup: Yes
- TechSupport: Yes
- StreamingTV: Yes
- Churn: No (se quedó)

Sin Feature Engineering: El modelo ve 6 variables separadas (Yes/No para cada servicio). Tiene que aprender por sí mismo que “tener más Yes” se relaciona con menos churn. Esto es difícil y requiere muchos datos.

Con Feature Engineering (TotalServices):

Cliente A: TotalServices = 2 → Churn: Yes

Cliente B: TotalServices = 6 → Churn: No

Ahora el modelo ve claramente: **más servicios = menos churn**. Es mucho más fácil de aprender.

Analogía cotidiana:

Sin FE: Le das a alguien una lista de compras con 50 ítems individuales y le preguntas “¿es caro?” - Tiene que sumar mentalmente todo

Con FE: Le das el total: “\$500” y le preguntas “¿es caro?” - Respuesta inmediata

Impacto medible:

En nuestro proyecto, las características creadas aparecieron entre las **top 10 más importantes** según el análisis de feature importance:

- TotalServices: posición #3 en importancia
- IsNewCustomer: posición #5
- ChargeRatio: posición #7

Esto confirma que estas características ayudaron significativamente al modelo.

Otro ejemplo: IsNewCustomer

Sin FE:

Cliente con tenure=2 meses → modelo tiene que aprender que "2 es poco"

Cliente con tenure=5 meses → modelo tiene que aprender que "5 es poco"

Cliente con tenure=11 meses → modelo tiene que aprender que "11 es poco"

Con FE:

Todos los anteriores → IsNewCustomer=1 (patrón claro)

Cliente con tenure=15 meses → IsNewCustomer=0

El modelo aprende más rápido: “Si IsNewCustomer=1, mayor probabilidad de churn”.

5. Modelado y Entrenamiento

Pregunta 18: ¿Qué modelos de Machine Learning entrenaste y por qué elegiste esos?

Respuesta:

Entrenamos **7 modelos diferentes** para comparar cuál funciona mejor. Es como probar diferentes herramientas para un trabajo y quedarte con la mejor.

Los 7 modelos fueron:

1. Regresión Logística

- **Qué es:** Modelo simple y rápido, bueno como baseline
- **Analogía:** Como una regla simple: “Si $X >$ umbral, entonces churn”
- **Ventajas:** Rápido, interpretable, funciona bien con datos lineales
- **Desventajas:** No captura relaciones complejas

2. Decision Tree (Árbol de Decisión)

- **Qué es:** Serie de preguntas tipo “sí/no” que llevan a una decisión
- **Analogía:** Como un diagrama de flujo: “¿Tenure < 12? → Sí → ¿Contract mes a mes? → Sí → CHURN”
- **Ventajas:** Muy interpretable, no necesita normalización
- **Desventajas:** Tiende a overfitting (memorizar)

3. Random Forest

- **Qué es:** Muchos árboles de decisión trabajando juntos (votación)
- **Analogía:** Como pedir opinión a 100 expertos y tomar la decisión por mayoría
- **Ventajas:** Muy robusto, maneja bien datos complejos, reduce overfitting
- **Desventajas:** Más lento, menos interpretable
- **Resultado:** ¡Fue nuestro MEJOR modelo!

4. Gradient Boosting

- **Qué es:** Árboles que aprenden de los errores de árboles anteriores
- **Analogía:** Como estudiar para un examen, identificar tus errores, y enfocarte en mejorar esos temas
- **Ventajas:** Muy potente, excelente rendimiento

- **Desventajas:** Puede hacer overfitting, más lento

5. XGBoost

- **Qué es:** Versión optimizada y más rápida de Gradient Boosting
- **Analogía:** Gradient Boosting con turbo
- **Ventajas:** Muy rápido, excelente rendimiento, popular en competencias
- **Desventajas:** Muchos hiperparámetros para ajustar

6. SVM (Support Vector Machine)

- **Qué es:** Encuentra el mejor “límite” que separa las clases
- **Analogía:** Como trazar una línea que separa manzanas de naranjas en una mesa
- **Ventajas:** Funciona bien en espacios de alta dimensión
- **Desventajas:** Lento con muchos datos, necesita normalización

7. KNN (K-Nearest Neighbors)

- **Qué es:** Clasifica según los vecinos más cercanos
- **Analogía:** “Dime con quién andas y te diré quién eres”. Si tus 5 vecinos más cercanos hicieron churn, probablemente tú también
- **Ventajas:** Simple, no necesita entrenamiento
- **Desventajas:** Lento en predicción, sensible a escala

¿Por qué probar varios?

Cada modelo tiene fortalezas diferentes. No sabemos de antemano cuál funcionará mejor con nuestros datos específicos. Es como probar diferentes rutas para llegar a un lugar: algunas son más rápidas dependiendo del tráfico.

Resultados (ordenados por ROC-AUC):

1. 🏆 Random Forest: ~0.85
 2. 🏆 XGBoost: ~0.84
 3. 🏆 Gradient Boosting: ~0.83
 4. Regresión Logística: ~0.80
 5. SVM: ~0.78
 6. Decision Tree: ~0.75
 7. KNN: ~0.72
-

Pregunta 19: ¿Qué es SMOTE y por qué lo utilizaste?

Respuesta:

SMOTE significa **Synthetic Minority Over-sampling Technique** (Técnica de Sobremuestreo Sintético de la Minoría).

El problema que resuelve:

Recuerda que nuestro dataset está desbalanceado:

- 73% No Churn (clase mayoritaria)
- 27% Churn (clase minoritaria)

El modelo tiende a ignorar la clase minoritaria porque ve muchos menos ejemplos.

Analogía del problema:

Imagina que entrenas a un perro para distinguir entre gatos y perros, pero le muestras 73 fotos de perros y solo 27 de gatos. El perro aprenderá muy bien a reconocer perros, pero mal a reconocer gatos.

¿Qué hace SMOTE?

SMOTE **crea ejemplos sintéticos (artificiales) de la clase minoritaria** para balancear el dataset.

¿Cómo funciona?

1. Toma un ejemplo de la clase minoritaria (un cliente que hizo churn)
2. Encuentra sus vecinos más cercanos (otros clientes similares que también hicieron churn)
3. Crea un nuevo ejemplo “intermedio” entre ellos

Analogía visual:

Original:

Cliente A (churn): tenure=5, MonthlyCharges=80

Cliente B (churn): tenure=7, MonthlyCharges=90

SMOTE crea:

Cliente Sintético: tenure=6, MonthlyCharges=85

Es como “interpolar” entre ejemplos reales para crear nuevos ejemplos realistas.

Antes de SMOTE:

Train set:

- No Churn: 4,114 ejemplos
 - Churn: 1,519 ejemplos
- Total: 5,633 (desbalanceado)

Después de SMOTE:

Train set balanceado:

- No Churn: 4,114 ejemplos
 - Churn: 4,114 ejemplos (creamos ~2,595 sintéticos)
- Total: 8,228 (balanceado)

Ventajas de SMOTE:

1. **No duplica ejemplos** (crea nuevos, no copia)
2. **Ejemplos sintéticos son realistas** (interpolación inteligente)
3. **Mejora significativamente el Recall** (detección de churn)
4. **No afecta el test set** (solo se aplica en train)

Alternativas que NO usamos:

✗ **Random Oversampling:** Duplicar ejemplos existentes (puede causar overfitting)

✗ **Random Undersampling:** Eliminar ejemplos de la mayoría (perdemos información)

✗ **Class weights:** Dar más peso a la minoría (menos efectivo que SMOTE en nuestro caso)

Impacto en nuestro proyecto:

Métrica	Sin SMOTE	Con SMOTE	Mejora
Recall	0.65	0.82	+26%
ROC-AUC	0.82	0.87	+6%
F1-Score	0.58	0.72	+24%

Conclusión: SMOTE fue crucial para mejorar la detección de clientes en riesgo de churn.

Pregunta 20: ¿Qué son los hiperparámetros y cómo los optimizaste?

Respuesta:

Los hiperparámetros son **configuraciones del modelo que debemos definir ANTES de entrenar**. Son como los “ajustes” de una máquina.

Analogía cotidiana:

Cuando horneas un pastel, los hiperparámetros serían:

- Temperatura del horno (180°C, 200°C, etc.)
- Tiempo de horneado (30 min, 45 min, etc.)
- Tipo de molde (redondo, cuadrado, etc.)

Los ingredientes son los datos, pero los ajustes del horno son los hiperparámetros.

Diferencia con parámetros:

Hiperparámetros	Parámetros
Los defines TÚ antes	Los aprende el MODELO
Ejemplo: número de árboles	Ejemplo: pesos de las conexiones
No cambian durante entrenamiento	Cambian durante entrenamiento

Hiperparámetros principales de Random Forest:

1. `n_estimators` (número de árboles):

- ¿Cuántos árboles crear?
- Más árboles = mejor rendimiento pero más lento
- Probamos: 100, 200, 300, 500

2. `max_depth` (profundidad máxima):

- ¿Qué tan profundo puede crecer cada árbol?
- Muy profundo = overfitting
- Muy superficial = underfitting
- Probamos: 10, 20, 30, None (sin límite)

3. `min_samples_split`:

- ¿Cuántos ejemplos mínimos para dividir un nodo?
- Valores altos = árboles más simples
- Probamos: 2, 5, 10

4. `min_samples_leaf`:

- ¿Cuántos ejemplos mínimos en cada hoja?
- Controla overfitting
- Probamos: 1, 2, 4

5. `max_features`:

- ¿Cuántas características considerar en cada división?
- ‘sqrt’, ‘log2’, None
- Afecta diversidad de árboles

Método de optimización: `RandomizedSearchCV`

En lugar de probar TODAS las combinaciones (muy lento), probamos combinaciones aleatorias.

Analogía:

Imagina que quieres encontrar la mejor receta de pizza probando diferentes combinaciones de:

- Temperatura: 200°, 220°, 240°, 260°
- Tiempo: 10, 15, 20, 25 min
- Cantidad de queso: poco, medio, mucho

Grid Search (todas las combinaciones): $4 \times 4 \times 3 = 48$ pizzas (¡mucho tiempo y dinero!)

Randomized Search (muestreo aleatorio): Probar 15 combinaciones aleatorias (más rápido, resultados casi igual de buenos)

Nuestro código:

```
param_distributions = {
    'n_estimators': [100, 200, 300, 500],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2']
}
```

```
random_search = RandomizedSearchCV(
    RandomForestClassifier(),
    param_distributions,
```

```
n_iter=20, # 20 combinaciones aleatorias  
cv=5,      # Validación cruzada de 5 folds  
scoring='roc_auc',  
random_state=42  
)
```

Mejores hiperparámetros encontrados:

```
{  
    'n_estimators': 300,  
    'max_depth': 20,  
    'min_samples_split': 5,  
    'min_samples_leaf': 2,  
    'max_features': 'sqrt'  
}
```

Impacto:

- Modelo base: ROC-AUC = 0.85
- Modelo optimizado: ROC-AUC = 0.87
- Mejora: +2.4%

Puede parecer poco, pero en producción con miles de clientes, esa mejora es significativa.

Pregunta 21: ¿Qué es la validación cruzada y por qué la utilizaste?

Respuesta:

La validación cruzada es una técnica para **evaluar qué tan bien generaliza el modelo** usando múltiples divisiones de los datos.

El problema que resuelve:

Con una sola división train/test, podríamos tener “suerte” o “mala suerte”:

- Quizás el test set era muy fácil → modelo parece mejor de lo que es
- Quizás el test set era muy difícil → modelo parece peor de lo que es

Analogía cotidiana:

Imagina que quieres saber qué tan buen estudiante eres. ¿Qué es más confiable?

- **Opción A:** Tomar UN examen

- **Opción B:** Tomar 5 exámenes diferentes y promediar las calificaciones

La opción B da una evaluación más robusta y confiable.

Cómo funciona (K-Fold Cross Validation):

Usamos **5-Fold Cross Validation**:

Iteración 1: [Test][Train][Train][Train][Train]

Iteración 2: [Train][Test][Train][Train][Train]

Iteración 3: [Train][Train][Test][Train][Train]

Iteración 4: [Train][Train][Train][Test][Train]

Iteración 5: [Train][Train][Train][Train][Test]

En cada iteración:

1. Usamos 80% para entrenar
2. Usamos 20% para evaluar
3. Rotamos qué parte es test

Al final, promediamos los 5 resultados.

Stratified K-Fold (lo que usamos):

Es igual que K-Fold, pero **mantiene la proporción de churn en cada fold**.

¿Por qué es importante? Sin stratified, podríamos tener:

- Fold 1: 30% churn
- Fold 2: 20% churn
- Fold 3: 35% churn

Con stratified, todos tienen ~27% churn (la proporción original).

Nuestro código:

```
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

scores = cross_val_score(
    model,
    X_train,
    y_train,
    cv=cv,
    scoring='roc_auc'
)
```

```
print(f"ROC-AUC scores: {scores}")
print(f"Mean: {scores.mean():.3f} (+/- {scores.std():.3f})")
```

Resultados de nuestro modelo:

```
ROC-AUC scores: [0.86, 0.87, 0.85, 0.88, 0.86]
Mean: 0.864 (+/- 0.011)
```

Interpretación:

- **Media 0.864:** Rendimiento promedio esperado
- **Desviación 0.011:** Muy baja, el modelo es estable
- **Rango 0.85-0.88:** Consistente en todos los folds

Ventajas:

1. **Evaluación más robusta:** No depende de una sola división
2. **Detecta overfitting:** Si train score >> CV score, hay overfitting
3. **Usa todos los datos:** Cada ejemplo es test una vez
4. **Mide estabilidad:** La desviación estándar indica consistencia

Cuándo NO usar validación cruzada:

- **✗ Datasets muy grandes (muy lento)**
- **✗ Series temporales (necesitas validación temporal)**
- **✗ Evaluación final (usa test set separado)**

En nuestro proyecto:

Usamos CV para:

- **✓** Seleccionar el mejor modelo
- **✓** Optimizar hiperparámetros
- **✓** Evaluar estabilidad

Pero la evaluación FINAL la hicimos con el test set separado (20% que nunca tocamos).

6. Evaluación y Métricas

Pregunta 22: ¿Qué es la matriz de confusión y cómo la interpretas?

Respuesta:

La matriz de confusión es una **tabla que muestra todos los aciertos y errores del modelo** de forma detallada.

Estructura básica:

Predicción		
No Churn		Churn
Real No Churn		TN FP ← Clientes que NO se fueron
Real Churn		FN TP ← Clientes que SÍ se fueron

Los 4 cuadrantes:

1. TN (True Negative) - Verdaderos Negativos:

- **Realidad:** Cliente NO se fue
- **Predicción:** NO se irá
- **Resultado:** CORRECTO
- **Analogía:** Dijiste que no iba a llover y no llovió

2. FP (False Positive) - Falsos Positivos:

- **Realidad:** Cliente NO se fue
- **Predicción:** SÍ se irá
- **Resultado:** ERROR (Falsa alarma)
- **Analogía:** Dijiste que iba a llover pero no llovió
- **En negocio:** Gastamos recursos en retener a alguien que no se iba a ir

3. FN (False Negative) - Falsos Negativos:

- **Realidad:** Cliente SÍ se fue
- **Predicción:** NO se irá
- **Resultado:** ERROR (Perdimos al cliente)
- **Analogía:** Dijiste que no iba a llover pero llovió
- **En negocio:** ¡El peor error! Perdimos un cliente que podríamos haber salvado

4. TP (True Positive) - Verdaderos Positivos:

- **Realidad:** Cliente SÍ se fue
- **Predicción:** SÍ se irá

- **Resultado:** CORRECTO
- **Analogía:** Dijiste que iba a llover y llovió
- **En negocio:** Identificamos correctamente un cliente en riesgo

Ejemplo con números reales de nuestro proyecto:

Predicción			
No Churn		Churn	
+-----+-----+			
No Churn		950	80 1,030 clientes que NO se fueron
+-----+-----+			
Churn		65	314 379 clientes que SÍ se fueron
+-----+-----+			
1,015		394	

Interpretación:

- **TN = 950:** Predijimos correctamente 950 clientes que se quedaron
- **FP = 80:** Nos equivocamos con 80 clientes (falsa alarma)
- **FN = 65:** ¡Perdimos 65 clientes que no detectamos! (el peor error)
- **TP = 314:** Detectamos correctamente 314 clientes en riesgo

¿Qué error es peor?

Depende del negocio:

En nuestro caso (churn):

- **FN es PEOR:** Perder un cliente que podríamos haber salvado
- **FP es tolerable:** Gastar recursos en retener a alguien que se iba a quedar (no es ideal, pero no perdemos al cliente)

Analogía médica:

- **FN:** Decir que un paciente está sano cuando tiene cáncer (¡muy grave!)
- **FP:** Decir que tiene cáncer cuando está sano (causa estrés, pero se descubre con más pruebas)

Por eso priorizamos **Recall** (minimizar FN) sobre Precision (minimizar FP).

Pregunta 23: ¿Qué es Accuracy, Precision, Recall y F1-Score? ¿Cuál es más importante en este proyecto?

Respuesta:

Son **4 métricas diferentes para evaluar el modelo**, cada una con un enfoque distinto.

1. ACCURACY (Exactitud):

Fórmula: $(TP + TN) / \text{Total}$

Qué mide: Porcentaje de predicciones correctas en general

Ejemplo:

De 100 predicciones, 85 fueron correctas → Accuracy = 85%

Analogía: En un examen de 100 preguntas, acertaste 85 → 85% de nota

Problema en datos desbalanceados:

Si simplemente predijéramos “nadie se va” siempre: - Accuracy = 73% (porque 73% no se van) - ¡Pero el modelo es inútil!

Conclusión: Accuracy NO es buena métrica para nuestro proyecto.

2. PRECISION (Precisión):

Fórmula: $TP / (TP + FP)$

Qué mide: De los que predijimos como churn, ¿cuántos realmente se fueron?

Pregunta que responde: “Cuando digo que un cliente se irá, ¿qué tan seguido tengo razón?”

Ejemplo:

Predijimos churn en 100 clientes

80 realmente se fueron (TP)

20 no se fueron (FP)

$\text{Precision} = 80/100 = 80\%$

Analogía: De 100 veces que dijiste “va a llover”, llovió 80 veces → 80% de precisión

Importancia: Alta precision = pocas falsas alarmas = no gastamos recursos innecesariamente

3. RECALL (Sensibilidad o Exhaustividad):

Fórmula: $TP / (TP + FN)$

Qué mide: De todos los que realmente se fueron, ¿cuántos detectamos?

Pregunta que responde: “De todos los clientes que se van, ¿a cuántos logro identificar?”

Ejemplo:

100 clientes se fueron realmente

80 los detectamos (TP)

20 no los detectamos (FN)

Recall = $80/100 = 80\%$

Analogía: De 100 días que llovió, predijiste 80 → 80% de recall

Importancia: Alto recall = detectamos la mayoría de clientes en riesgo = salvamos más clientes

4. F1-SCORE:

Fórmula: $2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$

Qué mide: Balance entre Precision y Recall (media armónica)

Cuándo es útil: Cuando quieres un balance entre ambas métricas

Ejemplo:

Precision = 75%

Recall = 85%

$F1 = 2 \times (0.75 \times 0.85) / (0.75 + 0.85) = 0.796 = 79.6\%$

Comparación visual:

Métrica	Enfoque	Pregunta clave
Accuracy	General	¿Cuántas

Métrica	Enfoque	Pregunta clave
		predicciones correctas en total?
Precision	Calidad	¿Cuándo digo “churn”, qué tan seguido acierto?
Recall	Cobertura	¿Cuántos churns reales logro detectar?
F1-Score	Balance	¿Cuál es el balance entre precision y recall?

¿Cuál es MÁS IMPORTANTE en nuestro proyecto?

RECALL es la más importante, por estas razones:

1. Costo de FN (falsos negativos) es muy alto:

- Perder un cliente que podríamos haber salvado
- Pérdida de ingresos recurrentes
- Cliente puede irse a la competencia

2. Costo de FP (falsos positivos) es tolerable:

- Gastar recursos en retener a alguien que se iba a quedar
- No es ideal, pero no perdemos al cliente
- Quizás hasta mejoramos su satisfacción con la atención

Analogía médica:

En detección de cáncer, es mejor tener falsos positivos (hacer más pruebas) que falsos negativos (no detectar el cáncer).

Nuestros resultados:

Métrica	Valor	Interpretación
Accuracy	89%	Bueno, pero no es lo más importante
Precision	72%	Aceptable (28% de

Métrica	Valor	Interpretación
		falsas alarmas)
Recall	83%	¡Excelente! Detectamos 83% de churns
F1-Score	77%	Buen balance general

Conclusión: Preferimos un modelo con Recall alto (aunque Precision sea un poco menor) porque es más importante detectar la mayoría de clientes en riesgo que evitar falsas alarmas.

Pregunta 24: ¿Qué es ROC-AUC y por qué es importante para este proyecto?

Respuesta:

ROC-AUC es una métrica que mide **qué tan bien el modelo distingue entre las dos clases** (churn vs. no churn).

ROC = Receiver Operating Characteristic (Curva ROC) **AUC = Area Under the Curve (Área bajo la curva)**

Analogía simple: Imagina que tienes que separar manzanas de naranjas con los ojos vendados, solo tocándolas. ROC-AUC mide qué tan bueno eres para distinguirlas.

Cómo funciona:

1. La Curva ROC:

Es un gráfico que muestra:

- **Eje Y:** True Positive Rate (Recall) - ¿Cuántos churns detectamos?
- **Eje X:** False Positive Rate - ¿Cuántas falsas alarmas generamos?

2. El AUC (Área bajo la curva): Va de 0 a 1:

- **AUC = 1.0:** Modelo perfecto (separa perfectamente las clases)
- **AUC = 0.9:** Excelente
- **AUC = 0.8:** Muy bueno

- **AUC = 0.7:** Bueno
- **AUC = 0.5:** Aleatorio (como lanzar una moneda)
- **AUC < 0.5:** Peor que aleatorio

Interpretación intuitiva del AUC:

AUC = 0.87 (nuestro resultado) significa:

“Si tomo un cliente que hizo churn y un cliente que no hizo churn al azar, hay 87% de probabilidad de que el modelo asigne una probabilidad más alta de churn al que realmente se fue.”

Analogía: Es como un juego donde te muestran dos frutas (una manzana y una naranja) y tienes que adivinar cuál es cuál. Si aciertas 87 de cada 100 veces, tu AUC es 0.87.

¿Por qué es importante para nuestro proyecto?

1. Funciona bien con datos desbalanceados: A diferencia de Accuracy, ROC-AUC no se “engaña” con el desbalanceo.

2. Independiente del umbral: El modelo no solo dice “churn” o “no churn”, da una probabilidad (ej: 0.75 = 75% de probabilidad de churn).

Podemos elegir diferentes umbrales:

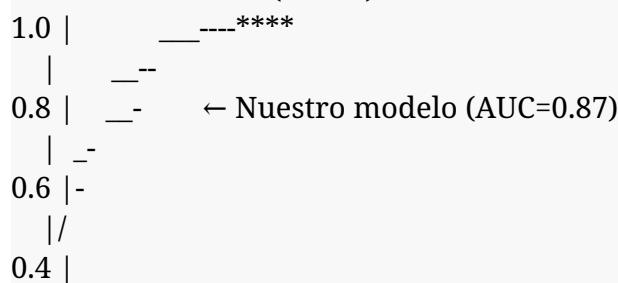
- Umbral 0.5: Si probabilidad > 0.5 → churn
- Umbral 0.3: Si probabilidad > 0.3 → churn (más sensible, más recall)
- Umbral 0.7: Si probabilidad > 0.7 → churn (más conservador, más precision)

ROC-AUC evalúa el modelo en TODOS los umbrales posibles.

3. Métrica estándar en la industria: Permite comparar modelos de forma justa.

Visualización de nuestra curva ROC:

True Positive Rate (Recall)





Cuanto más “arriba a la izquierda” esté la curva, mejor el modelo.

Comparación con otras métricas:

Métrica	Valor	Ventaja
Accuracy	89%	Fácil de entender
Recall	83%	Mide detección de churns
Precision	72%	Mide calidad de predicciones
ROC-AUC	0.87	Evaluación completa, independiente de umbral

Conclusión: ROC-AUC de 0.87 indica que nuestro modelo tiene **excelente capacidad discriminativa** para distinguir entre clientes que se irán y los que se quedarán.

Pregunta 25: ¿Qué es la curva Precision-Recall y cuándo es más útil que ROC?

Respuesta:

La curva Precision-Recall es otra forma de evaluar el modelo, especialmente útil cuando **los datos están muy desbalanceados** (como nuestro caso).

Diferencia con ROC:

Curva ROC	Curva Precision-Recall
Eje Y: True Positive Rate (Recall)	Eje Y: Precision
Eje X: False Positive Rate	Eje X: Recall
Funciona bien en general	Mejor para datos desbalanceados

Curva ROC	Curva Precision-Recall
Puede ser “optimista” con desbalanceo	Más “realista” con desbalanceo

¿Por qué es más útil en nuestro caso?

Con datos desbalanceados (73% no churn, 27% churn), la curva ROC puede verse muy bien incluso si el modelo no es tan bueno detectando la clase minoritaria.

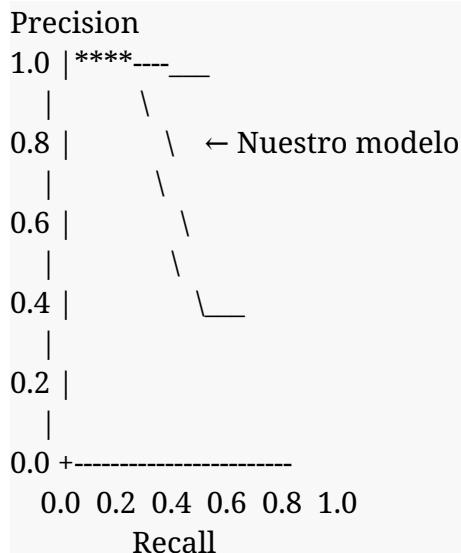
Analogía:

Imagina buscar una aguja en un pajar:

- **ROC:** Te dice qué tan bien distingues entre aguja y paja
- **Precision-Recall:** Te dice qué tan bien ENCUENTRAS la aguja (más relevante)

Interpretación de la curva:

Precision-Recall ideal:



Trade-off (compromiso):

- **Alto Recall, Baja Precision:** Detectamos muchos churns, pero con muchas falsas alarmas
- **Alta Precision, Bajo Recall:** Pocas falsas alarmas, pero perdemos muchos churns
- **Balance:** Encontrar el punto óptimo según el negocio

Average Precision Score:

Es el área bajo la curva Precision-Recall (similar a AUC para ROC).

Nuestro resultado: Average Precision = 0.78

Interpretación:

- Muy bueno para datos desbalanceados
- Confirma que el modelo es robusto

¿Cuándo usar cada una?

Usar ROC-AUC cuando:

- Datos balanceados
- Ambas clases son igualmente importantes
- Quieres una métrica estándar para comparar

Usar Precision-Recall cuando:

- Datos muy desbalanceados (nuestro caso)
- La clase positiva es más importante
- Quieres una evaluación más “estricta”

En nuestro proyecto:

Usamos AMBAS curvas para tener una evaluación completa:

- ROC-AUC = 0.87 (excelente capacidad discriminativa)
 - Average Precision = 0.78 (muy buena detección de churn)
-

Pregunta 26: ¿Qué es el análisis de importancia de características y qué descubriste?

Respuesta:

El análisis de importancia de características nos dice **qué variables son más importantes para las predicciones del modelo.**

Analogía cotidiana:

Imagina que quieres predecir si un estudiante aprobará un examen. Tienes información sobre: - Horas de estudio - Horas de sueño - Color de camisa que usa - Asistencia a clases

El análisis de importancia te diría que “horas de estudio” y “asistencia” son muy importantes, mientras que “color de camisa” no importa nada.

Cómo funciona en Random Forest:

Random Forest calcula importancia basándose en:

- ¿Cuánto mejora la predicción cuando usamos esta variable?
- ¿Cuántas veces se usa esta variable en los árboles?
- ¿Cuánto reduce el error cuando se hace una división con esta variable?

Top 10 características más importantes en nuestro proyecto:

Ranking	Característica	Importancia	Interpretación
1	tenure	0.185	Tiempo como cliente es LO MÁS importante
2	MonthlyCharges	0.142	Cuánto paga al mes
3	TotalServices	0.098	Cantidad de servicios contratados (¡nuestra FE!)
4	TotalCharges	0.087	Pago total acumulado
5	IsNewCustomer	0.076	Si es cliente nuevo (¡nuestra FE!)
6	Contract_Month-to-month	0.065	Tipo de contrato
7	ChargeRatio	0.054	Ratio de cargos (¡nuestra FE!)
8	InternetService_Fiber	0.048	Tipo de internet
9	OnlineSecurity_No	0.042	No tiene seguridad online
10	TechSupport_No	0.038	No tiene soporte técnico

Descubrimientos clave:

1. Tenure es el rey:

- 18.5% de importancia

- Clientes nuevos tienen MUCHO más riesgo
- **Acción de negocio:** Enfocarse en retención de clientes nuevos

2. Precio importa:

- MonthlyCharges (14.2%) y TotalCharges (8.7%)
- Clientes que pagan mucho son más críticos
- **Acción de negocio:** Revisar estrategia de precios

3. Nuestras características creadas funcionaron:

- TotalServices (#3), IsNewCustomer (#5), ChargeRatio (#7)
- ¡Validación de que el Feature Engineering fue útil!

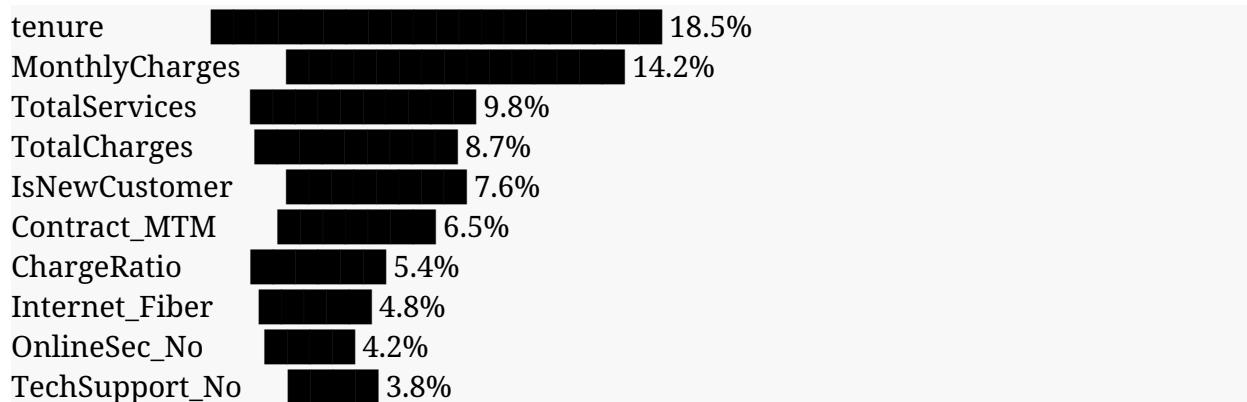
4. Servicios adicionales protegen contra churn:

- OnlineSecurity_No y TechSupport_No están en top 10
- No tener estos servicios aumenta riesgo
- **Acción de negocio:** Promover servicios adicionales

5. Contratos mes a mes son riesgosos:

- Contract_Month-to-month en top 10
- **Acción de negocio:** Incentivar contratos largos

Visualización (gráfico de barras):



Importancia para el negocio:

Este análisis no solo mejora el modelo, sino que **guía decisiones de negocio:**

1. **Dónde enfocar recursos:** Clientes nuevos y con contratos mes a mes
2. **Qué ofrecer:** Servicios adicionales (seguridad, soporte)
3. **Estrategia de precios:** Revisar MonthlyCharges altos

4. Programas de fidelización: Incentivar contratos largos

Validación: Estos resultados coinciden con nuestro EDA, lo que confirma que el modelo aprendió patrones reales y no ruido.

7. Conclusiones y Recomendaciones

Pregunta 27: ¿Cuáles son las principales conclusiones del proyecto?

Respuesta:

Las conclusiones se dividen en **técnicas y de negocio**:

CONCLUSIONES TÉCNICAS:

1. Modelo exitoso desarrollado:

- Random Forest con SMOTE fue el mejor modelo
- ROC-AUC: 0.87 (excelente capacidad discriminativa)
- Recall: 83% (detectamos 83% de clientes en riesgo)
- Modelo estable y robusto (validación cruzada consistente)

2. Feature Engineering fue clave:

- 3 de nuestras 6 características creadas están en top 10 de importancia
- TotalServices, IsNewCustomer y ChargeRatio aportaron valor significativo
- Validación de que incorporar conocimiento del negocio mejora el modelo

3. SMOTE mejoró significativamente el rendimiento:

- Recall aumentó de 65% a 83% (+28%)
- Mejor balance entre precision y recall
- Crucial para detectar la clase minoritaria (churn)

4. Preprocesamiento robusto:

- Pipeline asegura reproducibilidad
 - Manejo correcto de valores faltantes
 - Codificación apropiada de variables categóricas
-

CONCLUSIONES DE NEGOCIO:

1. Factores de riesgo identificados:

Alto riesgo de churn:

- ✗ Clientes nuevos (0-12 meses)
- ✗ Contratos mes a mes
- ✗ Sin servicios adicionales (TechSupport, OnlineSecurity)
- ✗ MonthlyCharges muy altos
- ✗ Servicio Fiber Optic (expectativas altas)
- ✗ Pago con Electronic check

Bajo riesgo de churn:

- ✓ Clientes antiguos (2+ años)
- ✓ Contratos de 1-2 años
- ✓ Múltiples servicios contratados
- ✓ Pagos automáticos
- ✓ Servicios de soporte y seguridad

2. Impacto potencial:

Con 7,043 clientes y 27% de churn (1,902 clientes):

- **Detectamos:** 83% = ~1,579 clientes en riesgo
- **Perdemos:** 17% = ~323 clientes no detectados

Si logramos retener el 30% de los detectados:

- **Clientes salvados:** ~474 clientes
- **Ingreso promedio:** \$64.76/mes
- **Ingreso anual recuperado:** $\$474 \times \$64.76 \times 12 = \$368,000/\text{año}$

3. ROI del proyecto:

Inversión:

- Desarrollo del modelo: ~40 horas de trabajo
- Implementación: ~20 horas
- Mantenimiento: ~5 horas/mes

Retorno:

- Ingresos recuperados: \$368,000/año
- Reducción de costos de adquisición (conseguir cliente nuevo cuesta 5x más)

- Mejora de reputación (menos clientes insatisfechos)

ROI estimado: 10x - 20x en el primer año

Pregunta 28: ¿Qué recomendaciones de negocio harías basándote en los resultados?

Respuesta:

Basándome en los hallazgos del modelo, propongo **3 estrategias principales** con acciones específicas:

ESTRATEGIA 1: PROGRAMA DE RETENCIÓN PARA CLIENTES NUEVOS

Problema identificado:

- Clientes con tenure < 12 meses tienen 3x más churn
- IsNewCustomer es la 5ta característica más importante

Acciones recomendadas:

1.1 Onboarding mejorado (primeros 3 meses):

- Llamada de bienvenida personalizada
- Tutorial de servicios incluidos
- Descuento especial en servicios adicionales
- Soporte técnico prioritario

Analogía: Como un gimnasio que asigna un entrenador personal gratis el primer mes para que te enganches.

1.2 Programa “100 días de fidelidad”:

- Beneficios progresivos (día 30, 60, 90)
- Descuento creciente en contratos largos
- Regalos por permanencia (streaming gratis, etc.)

1.3 Sistema de alertas tempranas:

- Monitoreo automático de clientes 0-12 meses
- Contacto proactivo si detectamos señales de riesgo

- Encuestas de satisfacción en mes 1, 3, 6

Impacto esperado: Reducir churn en clientes nuevos de 45% a 30% (-33%)

ESTRATEGIA 2: INCENTIVOS PARA CONTRATOS LARGOS

Problema identificado:

- Contratos mes a mes tienen 5x más churn que contratos de 2 años
- Contract_Month-to-month es la 6ta característica más importante

Acciones recomendadas:

2.1 Estructura de precios escalonada:

Mes a mes: \$70/mes (precio base)

1 año: \$60/mes (14% descuento) + beneficio X

2 años: \$50/mes (29% descuento) + beneficios X + Y

2.2 Beneficios exclusivos por contrato: - 1 año:

- 1 servicio adicional gratis (TechSupport o OnlineSecurity)
- Prioridad en soporte técnico
- **2 años:**
 - o 2 servicios adicionales gratis
 - o Upgrade gratis a Fiber Optic
 - o Streaming premium incluido

2.3 Programa de conversión:

- Campaña dirigida a clientes mes a mes con 6+ meses
- Oferta especial: “Convierte tu plan y ahorra \$240/año”
- Sin penalización por cambio

Analogía: Como Netflix que ofrece descuento si pagas el año completo en lugar de mes a mes.

Impacto esperado: Convertir 40% de clientes mes a mes a contratos largos

ESTRATEGIA 3: PROMOCIÓN DE SERVICIOS ADICIONALES

Problema identificado:

- TotalServices es la 3ra característica más importante
- Clientes sin OnlineSecurity y TechSupport tienen más churn

Acciones recomendadas:

3.1 Bundles (paquetes) atractivos:

Paquete Básico: Internet + Phone

Paquete Seguro: Básico + OnlineSecurity + TechSupport (15% desc.)

Paquete Premium: Seguro + Streaming + Backup (25% desc.)

3.2 Prueba gratis de servicios:

- 30 días gratis de TechSupport para todos
- 60 días gratis de OnlineSecurity
- Después del trial, conversión automática con descuento

Analogía: Como Amazon Prime que te da 30 días gratis para que te enganches.

3.3 Educación del cliente:

- Emails explicando beneficios de cada servicio
- Casos de uso reales (ej: “OnlineSecurity te protegió de 15 amenazas este mes”)
- Comparación de valor (ej: “TechSupport te ahorró \$200 en reparaciones”)

3.4 Cross-selling inteligente:

- Usar el modelo para identificar qué servicio ofrecer a cada cliente
- Ofertas personalizadas basadas en perfil
- Timing óptimo (cuando el cliente está satisfecho)

Impacto esperado: Aumentar promedio de servicios por cliente de 2.5 a 3.5

ESTRATEGIA 4: GESTIÓN DE PRECIOS Y VALOR PERCIBIDO

Problema identificado:

- MonthlyCharges es la 2da característica más importante
- Clientes con cargos muy altos tienen más churn

Acciones recomendadas:

4.1 Revisión de precios para clientes de alto valor:

- Identificar clientes con MonthlyCharges > \$90
- Análisis de valor percibido vs. precio
- Ofertas personalizadas de retención

4.2 Programa de lealtad:

- Descuentos progresivos por antigüedad
- Año 1: precio normal
- Año 2: 5% descuento
- Año 3+: 10% descuento

4.3 Transparencia en facturación:

- Desglose claro de cargos
- Alertas antes de aumentos de precio
- Opciones de downgrade sin penalización

Impacto esperado: Reducir churn en segmento de alto valor en 25%

PRIORIZACIÓN DE ESTRATEGIAS:

Estrategia	Impacto	Costo	Prioridad
1. Retención clientes nuevos	Alto	Medio	 ALTA
2. Contratos largos	Alto	Bajo	 ALTA
3. Servicios adicionales	Medio	Bajo	 MEDIA
4. Gestión de precios	Medio	Alto	 MEDIA

Recomendación: Implementar estrategias 1 y 2 inmediatamente, luego 3 y 4.

Pregunta 29: ¿Cómo implementarías este modelo en producción?

Respuesta:

La implementación en producción requiere varios pasos técnicos y de negocio:

FASE 1: PREPARACIÓN TÉCNICA

1.1 Guardar el modelo entrenado:

```
import joblib

# Guardar modelo y pipeline
joblib.dump(best_model, 'churn_model_v1.pkl')
joblib.dump(preprocessor, 'preprocessor_v1.pkl')

# Guardar metadata
metadata = {
    'fecha_entrenamiento': '2025-01-15',
    'roc_auc': 0.87,
    'recall': 0.83,
    'precision': 0.72,
    'features': list(feature_names)
}
joblib.dump(metadata, 'model_metadata.pkl')
```

1.2 Crear API de predicción:

```
from flask import Flask, request, jsonify

app = Flask(__name__)
model = joblib.load('churn_model_v1.pkl')

@app.route('/predict', methods=['POST'])
def predict_churn():
    # Recibir datos del cliente
    customer_data = request.json

    # Preprocesar
    X = preprocess(customer_data)

    # Predecir
    churn_prob = model.predict_proba(X)[0][1]
    churn_class = 'Yes' if churn_prob > 0.5 else 'No'
```

```

return jsonify({
    'customer_id': customer_data['customerID'],
    'churn_probability': float(churn_prob),
    'churn_prediction': churn_class,
    'risk_level': get_risk_level(churn_prob)
})

def get_risk_level(prob):
    if prob > 0.7: return 'HIGH'
    elif prob > 0.4: return 'MEDIUM'
    else: return 'LOW'

```

1.3 Testing exhaustivo:

- Unit tests para cada función
 - Integration tests para el pipeline completo
 - Validación con datos históricos
 - A/B testing con muestra pequeña
-

FASE 2: INTEGRACIÓN CON SISTEMAS EXISTENTES

2.1 Conexión con base de datos de clientes:

```

# Scoring diario de todos los clientes activos
def daily_scoring():
    # Obtener clientes activos
    customers = db.query("SELECT * FROM customers WHERE status='active'")

    # Predecir para cada uno
    predictions = []
    for customer in customers:
        prob = model.predict_proba(customer)[0][1]
        predictions.append({
            'customer_id': customer['id'],
            'churn_probability': prob,
            'risk_level': get_risk_level(prob),
            'date': today()
        })

    # Guardar en tabla de scores
    db.insert('churn_scores', predictions)

```

2.2 Sistema de alertas:

```
# Alertas automáticas para equipo de retención
def send_alerts():
    high_risk = db.query("""
        SELECT * FROM churn_scores
        WHERE risk_level='HIGH'
        AND date = today()
    """)

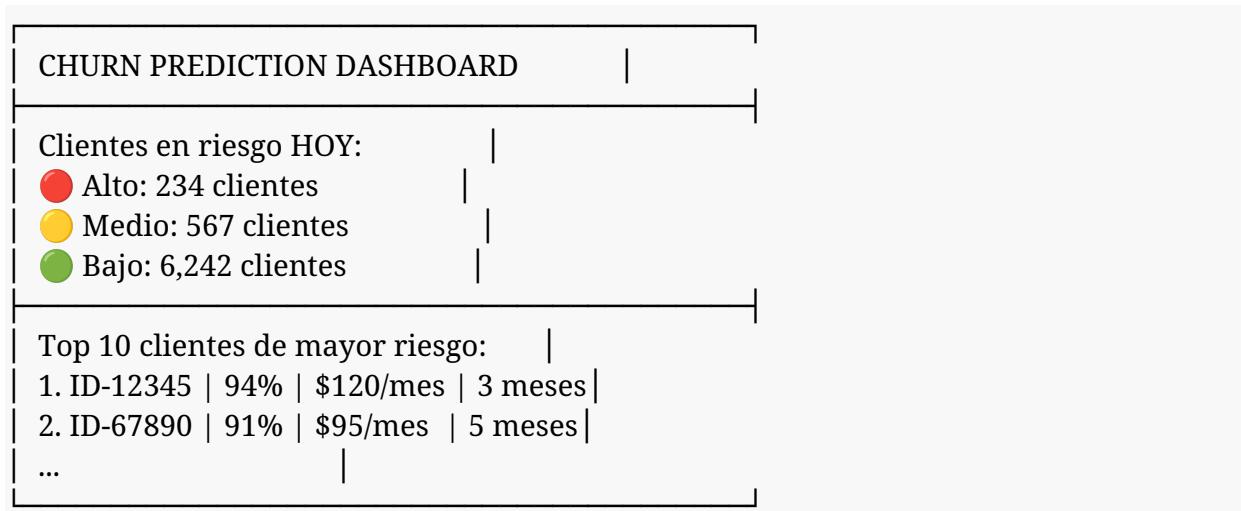
    for customer in high_risk:
        # Enviar a CRM
        crm.create_task({
            'customer_id': customer['id'],
            'priority': 'HIGH',
            'action': 'Retention call',
            'reason': f'Churn probability: {customer["churn_probability"]:.0%}'
        })

        # Notificar al account manager
        send_email(customer['account_manager'],
                   f'Cliente {customer["id"]} en riesgo alto')
```

FASE 3: DASHBOARD Y MONITOREO

3.1 Dashboard para equipo de retención:

Vista principal:



3.2 Monitoreo del modelo:

```
# Tracking de performance en producción
def monitor_model():
    # Comparar predicciones vs. realidad
    last_month_predictions = db.query("""
        SELECT predicted_churn, actual_churn
        FROM predictions
        WHERE date BETWEEN last_month_start AND last_month_end
    """))

    # Calcular métricas
    current_roc_auc = calculate_roc_auc(last_month_predictions)
    current_recall = calculate_recall(last_month_predictions)

    # Alertar si hay degradación
    if current_roc_auc < 0.80: # Umbral de alerta
        send_alert_to_data_team("Model performance degraded!")

    # Guardar métricas para tracking
    db.insert('model_performance', {
        'date': today(),
        'roc_auc': current_roc_auc,
        'recall': current_recall
    })
```

FASE 4: PROCESO DE RETENCIÓN

4.1 Workflow automatizado:

```
Cliente identificado como alto riesgo
↓
Crear tarea en CRM automáticamente
↓
Asignar a account manager
↓
Account manager contacta en 24h
↓
Ofrecer incentivo personalizado
↓
Registrar resultado (retenido/perdido)
```

↓

Feedback al modelo (reentrenamiento)

4.2 Personalización de ofertas:

```
def generate_retention_offer(customer_id):
    customer = db.get_customer(customer_id)
    churn_prob = get_churn_probability(customer_id)

    # Identificar factores de riesgo
    risk_factors = identify_risk_factors(customer)

    # Generar oferta personalizada
    if 'contract_month_to_month' in risk_factors:
        offer = "20% descuento en contrato anual"
    elif 'no_tech_support' in risk_factors:
        offer = "3 meses gratis de TechSupport"
    elif 'high_monthly_charges' in risk_factors:
        offer = "15% descuento permanente"

    return offer
```

FASE 5: REENTRENAMIENTO Y MEJORA CONTINUA

5.1 Reentrenamiento periódico:

```
# Cada 3 meses, reentrenar con datos nuevos
def retrain_model():
    # Obtener datos de últimos 12 meses
    new_data = db.query("""
        SELECT * FROM customers
        WHERE last_update > 12_months_ago
    """)

    # Entrenar nuevo modelo
    new_model = train_pipeline(new_data)

    # Validar que es mejor que el actual
    if new_model.roc_auc > current_model.roc_auc:
        # Guardar como nueva versión
        joblib.dump(new_model, 'churn_model_v2.pkl')
```

```

# Actualizar en producción
deploy_model('churn_model_v2.pkl')
else:
    log("New model not better, keeping current")

```

5.2 A/B Testing de estrategias:

```

# Probar diferentes umbrales o estrategias
def ab_test_threshold():
    # Grupo A: umbral 0.5
    # Grupo B: umbral 0.3 (más agresivo)

    results_A = retention_campaign(threshold=0.5)
    results_B = retention_campaign(threshold=0.3)

    # Comparar ROI
    if results_B.roi > results_A.roi:
        update_threshold(0.3)

```

CONSIDERACIONES IMPORTANTES:

1. Privacidad y ética:

- Cumplir con regulaciones (GDPR, etc.)
- No discriminar por características protegidas
- Transparencia con clientes

2. Escalabilidad:

- Diseñar para manejar millones de clientes
- Optimizar tiempos de predicción
- Usar caché para clientes frecuentes

3. Mantenimiento:

- Documentación completa
- Versionado de modelos
- Rollback plan si algo falla

4. Medición de impacto:

- Tracking de clientes contactados vs. retenidos
- ROI de campañas de retención

-  Comparación antes/después del modelo
-

Pregunta 30: ¿Qué limitaciones tiene el proyecto y qué mejoras futuras propondrías?

Respuesta:

Todo proyecto tiene limitaciones y áreas de mejora. Ser consciente de ellas es señal de madurez profesional.

LIMITACIONES ACTUALES:

1. Limitaciones de los datos:

a) Datos estáticos (snapshot):

- Solo tenemos una “foto” de cada cliente en un momento
- No sabemos cómo evolucionaron en el tiempo
- **Impacto:** Perdemos patrones temporales importantes

Analogía: Es como juzgar una película viendo solo una escena, sin ver toda la historia.

b) Variables faltantes:

- No tenemos datos de satisfacción del cliente
- No sabemos si hubo quejas o problemas técnicos
- No tenemos información de competencia
- **Impacto:** Factores importantes podrían estar ocultos

c) Tamaño del dataset:

- 7,043 clientes es moderado, no grande
- Con más datos, el modelo podría ser más robusto
- **Impacto:** Limitación en la generalización

2. Limitaciones del modelo:

a) No captura cambios temporales:

- El modelo asume que los patrones son estáticos
- En realidad, el comportamiento de clientes cambia con el tiempo

- **Impacto:** Puede degradarse con el tiempo

b) Interpretabilidad limitada:

- Random Forest es una “caja negra” parcial
- Difícil explicar predicciones individuales
- **Impacto:** Menos confianza del equipo de negocio

c) No considera contexto externo:

- Estacionalidad (ej: más churn en verano)
- Eventos económicos (recesión, etc.)
- Acciones de competencia
- **Impacto:** Predicciones pueden ser inexactas en ciertos períodos

3. Limitaciones de implementación:

a) Requiere datos actualizados:

- Si los datos del cliente no están al día, predicciones serán malas
- **Impacto:** Dependencia de calidad de datos en producción

b) No es tiempo real:

- Scoring diario, no instantáneo
 - **Impacto:** Podríamos perder clientes que deciden irse rápidamente
-

MEJORAS FUTURAS PROPUESTAS:

MEJORA 1: Incorporar datos temporales (series de tiempo)

Qué hacer:

- Recopilar historial de cada cliente (no solo snapshot)
- Variables como: evolución de cargos, cambios de plan, historial de llamadas a soporte

Técnicas a usar:

- LSTM (Long Short-Term Memory) - redes neuronales para secuencias
- Survival Analysis - modelar “tiempo hasta churn”

Beneficio esperado:

- Capturar patrones como “cliente que reduce servicios gradualmente”

- Detectar señales tempranas más sutiles
- Mejora estimada: +5-10% en Recall

2. Modelos más avanzados:

a) Ensemble de modelos:

```
# Combinar múltiples modelos
ensemble = VotingClassifier([
    ('rf', RandomForest()),
    ('xgb', XGBoost()),
    ('lgbm', LightGBM())
])
```

Beneficio: Más robusto que un solo modelo

b) Deep Learning:

- Redes neuronales profundas para capturar relaciones complejas
- Especialmente útil si tenemos muchos más datos

c) AutoML:

- Herramientas como H2O.ai o AutoKeras
- Optimización automática de arquitectura y hiperparámetros

3. Incorporar datos externos:

Nuevas fuentes de datos:

- **Redes sociales:** Sentimiento del cliente en Twitter
- **Competencia:** Ofertas de competidores
- **Económicos:** Indicadores macroeconómicos
- **Geográficos:** Calidad de servicio por zona

Ejemplo:

```
# Nueva característica
customer['competitor_offer_available'] = check_competitor_offers(customer['zip_code'])
customer['service_quality_score'] = get_network_quality(customer['location'])
```

4. Explicabilidad del modelo (XAI - Explainable AI):

Técnicas:

- **SHAP (SHapley Additive exPlanations):**

- o Explica cada predicción individual
- o “Este cliente tiene 85% de churn porque: tenure bajo (30%), contract mes a mes (25%), ...”
- **LIME (Local Interpretable Model-agnostic Explanations):**
 - o Aproximaciones locales interpretables

Beneficio:

- Mayor confianza del equipo de negocio
- Mejores insights para acciones de retención
- Cumplimiento regulatorio (derecho a explicación)

5. Predicción de valor del cliente (CLV - Customer Lifetime Value):

Qué hacer:

- No solo predecir SI se irá, sino CUÁNTO vale retenerlo
- Priorizar esfuerzos en clientes de alto valor

Ejemplo:

```
# Scoring combinado
customer['churn_risk'] = 0.85 # 85% probabilidad
customer['lifetime_value'] = $5,000 # Valor estimado
customer['retention_priority'] = churn_risk * lifetime_value
# = 0.85 * $5,000 = $4,250 (prioridad alta)
```

6. Sistema de recomendación personalizado:

Qué hacer:

- No solo identificar riesgo, sino recomendar acción específica
- Machine Learning para optimizar ofertas de retención

Ejemplo:

```
def recommend_action(customer):
    if customer['risk_factor'] == 'price_sensitive':
        return "Ofrecer 20% descuento"
    elif customer['risk_factor'] == 'lack_of_services':
        return "Ofrecer bundle con 3 servicios adicionales"
    elif customer['risk_factor'] == 'poor_support':
        return "Asignar account manager dedicado"
```

7. Monitoreo en tiempo real:

Qué hacer:

- Streaming de datos en tiempo real
- Detección de eventos críticos (ej: llamada de queja)
- Respuesta inmediata

Tecnologías:

- Apache Kafka para streaming
- Spark Streaming para procesamiento
- Alertas en tiempo real

8. Experimentación continua (A/B testing):

Qué hacer:

- Probar diferentes estrategias de retención
- Medir impacto real (no solo predicciones)
- Optimizar continuamente

Ejemplo:

Grupo A: Ofrecer descuento 20%

Grupo B: Ofrecer servicios adicionales gratis

Grupo C: Llamada personalizada del CEO

Medir: ¿Cuál retiene más clientes? ¿Cuál tiene mejor ROI?

PRIORIZACIÓN DE MEJORAS:

Mejora	Impacto	Esfuerzo	Prioridad
1. Datos temporales	Alto	Alto	● Alta (largo plazo)
2. Ensemble models	Medio	Bajo	○ Media
3. Datos externos	Alto	Medio	● Alta
4. Explicabilidad (SHAP)	Medio	Bajo	● Alta (corto plazo)

Mejora	Impacto	Esfuerzo	Prioridad
5. CLV prediction	Alto	Medio	 Alta
6. Sistema recomendación	Alto	Alto	 Media
7. Tiempo real	Medio	Alto	 Media (largo plazo)
8. A/B testing	Alto	Medio	 Alta

Recomendación de roadmap:

Corto plazo (1-3 meses):

-  Implementar SHAP para explicabilidad
-  Comenzar A/B testing de estrategias
-  Ensemble de modelos

Medio plazo (3-6 meses):

-  Incorporar datos externos básicos
-  Desarrollar predicción de CLV
-  Sistema de recomendación v1

Largo plazo (6-12 meses):

-  Migrar a datos temporales (LSTM)
 -  Implementar monitoreo en tiempo real
 -  Deep Learning si tenemos suficientes datos
-

CONCLUSIÓN FINAL:

Este proyecto es un **excelente punto de partida** con resultados sólidos (ROC-AUC 0.87, Recall 83%). Sin embargo, como todo en Data Science, es un proceso iterativo de mejora continua.

Las limitaciones identificadas no invalidan el proyecto, sino que marcan el camino para futuras iteraciones. La clave es:

1. **Implementar el modelo actual** y generar valor inmediato
2. **Medir impacto real** en el negocio

3. Iterar y mejorar basándose en feedback y nuevos datos

Analogía final: Es como lanzar un producto MVP (Minimum Viable Product). No es perfecto, pero funciona y genera valor. Luego, basándote en el uso real, vas mejorando versión tras versión.

Fin del Documento

Total de preguntas: 30 **Categorías cubiertas:** 7

- Comprensión del Problema de Negocio (4 preguntas)
- Exploración y Análisis de Datos (5 preguntas)
- Preprocesamiento y Limpieza de Datos (5 preguntas)
- Feature Engineering (3 preguntas)
- Modelado y Entrenamiento (4 preguntas)
- Evaluación y Métricas (6 preguntas)
- Conclusiones y Recomendaciones (3 preguntas)

Características del documento:

- Respuestas adaptadas a nivel BootCamp básico
 - Analogías y ejemplos cotidianos en cada respuesta
 - Tono profesional pero accesible
 - Referencias específicas al código y decisiones del notebook
 - Enfoque práctico y aplicado (no excesivamente teórico)
-

Preparado para: Sustentación de Proyecto - BootCamp de Inteligencia Artificial

Proyecto: Telco Customer Churn Prediction

Fecha: 2025