# Preguntas de Sustentación - Proyecto de Predicción de Customer Churn

Análisis de Customer Churn

## Contents

# Preguntas de Sustentación - Proyecto de Predicción de Customer Churn

## Información del Proyecto

- **Dataset**: Telco Customer Churn (7,043 clientes)
- **Objetivo**: Predecir qué clientes abandonarán el servicio de telecomunicaciones

- **Mejor Modelo**: Logistic Regression Optimizado (ROC-AUC: 0.8503)
- **Técnica de Balanceo**: Undersampling (seleccionada automáticamente por mejor ROC-AUC)

---

# CATEGORÍA 1: EXPLORACIÓN Y COMPRENSIÓN DE DATOS

## Pregunta 1

**¿Qué es el "churn" y por qué es importante predecirlo en una empresa de telecomunicaciones?**

**Respuesta**: El churn (o tasa de abandono) representa el porcentaje de clientes que dejan de usar los servicios de una empresa en un período determinado. En el dataset analizado, el 26.5% de los clientes abandonaron el servicio. Predecirlo es crucial porque adquirir un nuevo cliente cuesta entre 5 y 25 veces más que retener uno existente. Con un modelo predictivo, la empresa puede identificar clientes en riesgo y aplicar estrategias de retención proactivas.

**Analogía**: Es como un médico que detecta síntomas tempranos de una enfermedad. Si identificas a tiempo que un paciente (cliente) está "enfermo" (insatisfecho), puedes aplicar un tratamiento (oferta de retención) antes de que sea demasiado tarde.

**Mini-glosario**:

- **Churn Rate**: Porcentaje de clientes que abandonan en un período
- **Customer Lifetime Value (LTV)**: Valor total que un cliente genera durante su relación con la empresa
- **Retención**: Estrategias para mantener a los clientes activos

---

## Pregunta 2

**¿Cuál es la distribución de la variable objetivo (Churn) en el dataset y qué problema representa?**

**Respuesta**: El dataset presenta un desbalance de clases: 73% de clientes No Churn vs 27% Churn (ratio 2.7:1). Este desbalance es problemático porque los algoritmos de ML tienden a favorecer la clase mayoritaria, prediciendo casi siempre "No Churn" y fallando en detectar los casos que realmente importan (los clientes que sí abandonarán).

**Analogía**: Imagina un detector de incendios que funciona el 99% del tiempo, pero falla justo cuando hay fuego real. Un modelo que predice siempre "No Churn" tendría 73% de accuracy, pero sería inútil para el negocio.

**Mini-glosario**:

- **Desbalance de clases**: Cuando una categoría tiene muchos más ejemplos que otra
- **Clase mayoritaria/minoritaria**: La categoría con más/menos ejemplos
- **Accuracy paradox**: Alta precisión general pero fallo en la clase importante

---

## Pregunta 3

**¿Qué variables del dataset son numéricas y cuáles son categóricas? ¿Por qué es importante distinguirlas?**

**Respuesta**:

- **Numéricas (3)**: tenure (meses de antigüedad),
  MonthlyCharges (cargo mensual),
  TotalCharges (cargo total)
- **Categóricas (17)**: gender (género),
  SeniorCitizen (ciudadano mayor/tercera edad),
  Partner (pareja),
  Dependents (dependientes),
  PhoneService (servicio telefónico),
  MultipleLines (líneas múltiples),
  InternetService (servicio de internet),
  OnlineSecurity (seguridad en línea),
  OnlineBackup (respaldo en línea),
  DeviceProtection (protección de dispositivos),
  TechSupport (soporte técnico),
  StreamingTV (streaming de TV),
  StreamingMovies (streaming de películas),
  Contract (tipo de contrato),
  PaperlessBilling (facturación sin papel),
  PaymentMethod (método de pago)

Es importante distinguirlas porque requieren diferentes técnicas de preprocesamiento: las numéricas necesitan escalado (StandardScaler) y las categóricas necesitan codificación (OneHotEncoder).

**Analogía**: Es como preparar ingredientes para cocinar: las verduras (categóricas) se cortan de una forma y las carnes (numéricas) de otra. No puedes aplicar el mismo proceso a todos.

**Mini-glosario**:

- **Variable numérica**: Datos que representan cantidades medibles
- **Variable categórica**: Datos que representan grupos o categorías
- **Preprocesamiento**: Transformación de datos antes del modelado

---

## Pregunta 4

**¿Qué patrones encontraste en el análisis exploratorio respecto a los clientes que abandonan?**

**Respuesta**: Los principales patrones identificados fueron:

1. **Tenure bajo**: Clientes con menos de 12 meses tienen mayor probabilidad de churn
2. **Contratos mes a mes**: 42% de churn vs 3% en contratos de 2 años
3. **Sin servicios adicionales**: Clientes sin OnlineSecurity (seguridad en línea), TechSupport (soporte técnico) tienen más churn
4. **Cargos mensuales altos**: Correlación positiva con el abandono
5. **Fiber optic**: Mayor churn que DSL o sin internet

**Analogía**: Es como analizar por qué los estudiantes abandonan una universidad: los de primer año (tenure bajo), sin beca (sin servicios adicionales) y con matrículas altas (cargos altos) tienen más probabilidad de desertar.

**Mini-glosario**:

- **EDA (Exploratory Data Analysis)**: Análisis inicial para entender los datos
- **Correlación**: Relación estadística entre dos variables
- **Patrón**: Tendencia recurrente en los datos

---

### Pregunta 5

**¿Cómo manejaste los valores faltantes en el dataset?**

**Respuesta**: El dataset tenía 11 valores faltantes en la columna TotalCharges, correspondientes a clientes nuevos (tenure=0) donde el cargo total era un espacio en blanco. Se convirtió la columna a numérico con `pd.to_numeric(errors='coerce')` y se imputaron los valores nulos con el valor de MonthlyCharges, ya que un cliente nuevo en su primer mes tendría un cargo total igual a su cargo mensual.

**Analogía**: Es como completar el "total pagado" de un cliente nuevo usando su primera factura mensual. Si su cargo mensual es $50, su cargo total inicial también será $50.

**Mini-glosario**:

- **Valores faltantes (NaN)**: Datos ausentes en el dataset
- **Imputación**: Técnica para rellenar valores faltantes
- **Coerción**: Forzar la conversión de un tipo de dato a otro

---

## CATEGORÍA 2: PREPROCESAMIENTO Y FEATURE ENGINEERING

### Pregunta 6

**¿Qué es Feature Engineering y qué nuevas características creaste?**

**Respuesta**: Feature Engineering es el proceso de crear nuevas variables a partir de las existentes para mejorar el poder predictivo del modelo. Se crearon 2 características:

1. **Charge_Ratio**: MonthlyCharges / TotalCharges (ratio de cargos)
2. **Total_Services**: Suma de todos los servicios contratados

Estas características capturan información que no estaba explícita en las variables originales.

**Analogía**: Es como un chef que combina ingredientes básicos para crear una salsa especial. Los ingredientes individuales son buenos, pero la combinación aporta un sabor único que mejora el plato.

**Mini-glosario**:

- **Feature**: Variable o característica usada para entrenar el modelo
- **Feature Engineering**: Creación de nuevas variables derivadas
- **Poder predictivo**: Capacidad de una variable para predecir el resultado

---

## Pregunta 7

**¿Por qué es necesario escalar las variables numéricas?**

**Respuesta**: El escalado es necesario porque las variables numéricas tienen diferentes rangos: tenure va de 0-72 meses, MonthlyCharges de 18-118 dólares, y TotalCharges de 0-8,684 dólares. Sin escalado, los algoritmos como SVM, KNN y redes neuronales darían más peso a las variables con valores más grandes. StandardScaler transforma cada variable para tener media 0 y desviación estándar 1.

**Analogía**: Es como convertir diferentes monedas a una sola para poder compararlas. No puedes sumar directamente 100 yenes con 50 euros; necesitas convertirlos a la misma escala.

**Mini-glosario**:

- **StandardScaler**: Técnica que normaliza datos a media 0 y std 1
- **Normalización**: Ajustar valores a un rango común
- **Rango**: Diferencia entre el valor máximo y mínimo

---

## Pregunta 8

**¿Qué es OneHotEncoder y por qué se usa para variables categóricas?**

**Respuesta**: OneHotEncoder convierte variables categóricas en columnas binarias (0/1). Por ejemplo, la variable "Contract" con valores {Month-to-month, One year, Two year} se convierte en 3 columnas: Contract_Month-to-month, Contract_One year, Contract_Two year. Esto es necesario porque los algoritmos de ML trabajan con números, no con texto.

**Analogía**: Es como traducir un menú de restaurante a pictogramas para que personas de cualquier idioma puedan entenderlo. Cada plato tiene su propio símbolo único.

**Mini-glosario**:

- **OneHotEncoder**: Técnica de codificación binaria para categorías
- **Variable dummy**: Columna binaria creada por OneHotEncoding
- **Codificación**: Transformación de texto a números

---

## Pregunta 9

**¿Qué es un Pipeline en scikit-learn y por qué lo usaste?**

**Respuesta**: Un Pipeline es una secuencia de transformaciones que se aplican automáticamente en orden. En el proyecto se usó ColumnTransformer con dos pipelines: uno para variables numéricas (StandardScaler) y otro para categóricas (OneHotEncoder). Esto garantiza que las mismas transformaciones se apliquen consistentemente en entrenamiento y predicción, evitando data leakage.

**Analogía**: Es como una línea de ensamblaje en una fábrica: cada estación hace una tarea específica en orden, y el producto final siempre pasa por el mismo proceso.

**Mini-glosario**:

- **Pipeline**: Cadena de transformaciones secuenciales
- **ColumnTransformer**: Aplica diferentes transformaciones a diferentes columnas
- **Data leakage**: Filtración de información del test al entrenamiento

---

## Pregunta 10

**¿Cómo dividiste los datos en entrenamiento y prueba?**

**Respuesta**: Se usó `train_test_split` con 80% para entrenamiento y 20% para prueba, con `stratify=y` para mantener la proporción de clases en ambos conjuntos, y `random_state=42` para reproducibilidad. Esto resultó en 5,634 muestras de entrenamiento y 1,409 de prueba.

**Analogía**: Es como dividir un mazo de cartas para un juego: necesitas que ambos jugadores tengan una proporción similar de cartas rojas y negras para que el juego sea justo.

**Mini-glosario**:

- **Train/Test split**: División de datos para entrenamiento y evaluación
- **Stratify**: Mantener proporciones de clases en la división
- **Random state**: Semilla para reproducibilidad

---

# CATEGORÍA 3: MANEJO DE DESBALANCE DE CLASES

## Pregunta 11

**¿Qué técnicas de balanceo de clases evaluaste y cuál fue la mejor?**

**Respuesta**: Se evaluaron 3 técnicas:

1. **Undersampling** : Reduce la clase mayoritaria (ROC-AUC: 0.8277, Recall: 77.01%)
2. **SMOTE + Tomek Links**: SMOTE + eliminación de ejemplos ambiguos (ROC-AUC: 0.8273)
3. **SMOTE**: Crea ejemplos sintéticos de la clase minoritaria (ROC-AUC: 0.8256)

**Undersampling fue seleccionado** por obtener el mejor ROC-AUC y el mejor Recall, siendo además la técnica más rápida (0.58s vs 1.78s de SMOTE+Tomek).

**Analogía**: Es como equilibrar un aula donde hay 73 estudiantes de un grupo y 27 de otro. Puedes: (1) clonar estudiantes del grupo pequeño (SMOTE), (2) clonar y remover los más confusos (SMOTE+Tomek), o (3) sacar estudiantes del grupo grande (Undersampling).

**Mini-glosario**:

- **SMOTE**: Synthetic Minority Over-sampling Technique
- **Oversampling**: Aumentar ejemplos de la clase minoritaria
- **Undersampling**: Reducir ejemplos de la clase mayoritaria

---

## Pregunta 12

**¿Cómo funciona SMOTE a nivel técnico?**

**Respuesta**: SMOTE (Synthetic Minority Over-sampling Technique) genera ejemplos sintéticos de la clase minoritaria mediante interpolación. Para cada ejemplo de la clase minoritaria: (1) encuentra sus k vecinos más cercanos, (2) selecciona aleatoriamente uno de esos vecinos, (3) crea un nuevo ejemplo en un punto aleatorio de la línea que conecta ambos. Esto aumenta la diversidad sin simplemente duplicar ejemplos.

**Analogía**: Es como crear nuevos colores mezclando dos colores existentes. Si tienes azul y verde, puedes crear turquesa, aguamarina, etc. No estás copiando colores, estás creando nuevos que están "entre" los originales.

**Mini-glosario**:

- **Interpolación**: Crear valores intermedios entre dos puntos
- **K-vecinos**: Los k ejemplos más cercanos a un punto
- **Ejemplo sintético**: Dato artificial creado por el algoritmo

---

## Pregunta 13

**¿Por qué no simplemente duplicar los ejemplos de la clase minoritaria?**

**Respuesta**: Duplicar ejemplos (oversampling simple) causa overfitting porque el modelo memoriza los ejemplos repetidos en lugar de aprender patrones generales. SMOTE evita esto creando ejemplos nuevos que son similares pero no idénticos a los originales, lo que ayuda al modelo a generalizar mejor a datos no vistos.

**Analogía**: Es como estudiar para un examen. Si solo memorizas las mismas 10 preguntas repetidas, fallarás cuando aparezca una pregunta nueva. Pero si estudias variaciones de esas preguntas, estarás mejor preparado.

**Mini-glosario**:

- **Overfitting**: Modelo que memoriza en lugar de generalizar
- **Generalización**: Capacidad de funcionar bien con datos nuevos
- **Varianza**: Sensibilidad del modelo a cambios en los datos

---

## Pregunta 14

**¿Cuándo se aplica el balanceo: antes o después de dividir los datos?**

**Respuesta**: El balanceo SIEMPRE se aplica DESPUÉS de dividir los datos, y SOLO al conjunto de entrenamiento. Aplicarlo antes causaría data leakage porque los ejemplos sintéticos del test podrían estar basados en ejemplos del train. En el proyecto, primero se hizo train_test_split y luego se aplicó SMOTE solo a X_train.

**Analogía**: Es como preparar un examen: no puedes usar las respuestas del examen final para estudiar. El conjunto de test debe permanecer "puro" y representar datos del mundo real.

**Mini-glosario**:

- **Data leakage**: Contaminación del test con información del train
- **Conjunto de validación**: Datos separados para evaluar el modelo
- **Integridad de datos**: Mantener la separación train/test

---

# CATEGORÍA 4: MODELADO Y ALGORITMOS

## Pregunta 15

**¿Qué algoritmos de Machine Learning evaluaste y por qué esos específicamente?**

**Respuesta**: Se evaluaron 7 algoritmos:

1. **Logistic Regression**: Baseline simple e interpretable
2. **Decision Tree**: Fácil de interpretar, captura no linealidades
3. **Random Forest**: Ensemble de árboles, reduce overfitting
4. **Gradient Boosting**: Ensemble secuencial, muy potente
5. **XGBoost**: Versión optimizada de Gradient Boosting
6. **SVM**: Bueno para espacios de alta dimensión
7. **KNN**: Simple, basado en similitud

Se eligieron para cubrir diferentes familias de algoritmos y encontrar el mejor para este problema específico.

**Analogía**: Es como probar diferentes herramientas para un trabajo: martillo, destornillador, llave inglesa. Cada una tiene sus fortalezas, y solo probándolas sabes cuál funciona mejor para tu tarea.

**Mini-glosario**:

- **Ensemble**: Combinación de múltiples modelos
- **Baseline**: Modelo simple de referencia
- **Hiperparámetro**: Configuración del algoritmo

---

## Pregunta 16

**¿Por qué Logistic Regression Optimizado fue el mejor modelo?**

**Respuesta**: Logistic Regression Optimizado obtuvo el mejor ROC-AUC (0.8503) después de la optimización porque: (1) es un modelo lineal robusto e interpretable, (2) con la técnica de balanceo Undersampling mejora significativamente el Recall (79.41%), (3) es computacionalmente eficiente, (4) los coeficientes permiten interpretar la importancia de cada feature, (5) mostró consistencia en la validación de robustez con múltiples semillas (ROC-AUC promedio: $0.8513 \pm 0.0065$).

**Analogía**: Es como un juez experimentado que, aunque usa reglas simples y claras, logra tomar las mejores decisiones cuando se le da la información correctamente balanceada.

**Mini-glosario**:

- **Boosting**: Técnica que combina modelos débiles secuencialmente
- **Residuos**: Errores que el modelo anterior no pudo predecir
- **Regularización**: Técnica para prevenir overfitting

---

## Pregunta 17

**¿Qué es la validación cruzada y por qué la usaste?**

**Respuesta**: La validación cruzada (CV) divide los datos de entrenamiento en k partes (folds), entrena k modelos usando k-1 partes y valida con la restante, rotando. Se usó StratifiedKFold con 5 folds para obtener una estimación más robusta del rendimiento. El CV score promedio fue 0.84, similar al score en test, indicando que el modelo generaliza bien.

**Analogía**: Es como un estudiante que practica con 5 exámenes de prueba diferentes antes del examen real. Si obtiene notas similares en todos, sabes que realmente aprendió y no solo memorizó un examen específico.

**Mini-glosario**:

- **Cross-validation**: Técnica de validación con múltiples divisiones
- **Fold**: Cada partición de los datos en CV
- **StratifiedKFold**: CV que mantiene proporciones de clases

---

## Pregunta 18

**¿Qué es RandomizedSearchCV y cómo lo usaste para optimizar hiperparámetros?**

**Respuesta**: RandomizedSearchCV busca la mejor combinación de hiperparámetros probando combinaciones aleatorias de un espacio definido. Se definieron rangos para parámetros como n_estimators (50-300), max_depth (3-10), learning_rate (0.01-0.3), etc. Se probaron 50 combinaciones con 5-fold CV, optimizando ROC-AUC. Esto es más eficiente que GridSearch cuando el espacio de búsqueda es grande.

**Analogía**: Es como buscar el mejor restaurante en una ciudad. GridSearch visitaría TODOS los restaurantes; RandomizedSearch visita una muestra aleatoria representativa y encuentra uno excelente en menos tiempo.

**Mini-glosario**:

- **Hiperparámetro**: Configuración externa del modelo
- **Grid Search**: Búsqueda exhaustiva de todas las combinaciones
- **Espacio de búsqueda**: Rango de valores a explorar

---

## Pregunta 19

**¿Cuáles fueron los mejores hiperparámetros encontrados para Logistic Regression?**

**Respuesta**: Los hiperparámetros óptimos encontrados fueron:

- C: valor óptimo de regularización inversa
- penalty: tipo de regularización (L1 o L2)
- solver: algoritmo de optimización (liblinear o saga)
- max_iter: iteraciones máximas para convergencia
- learning_rate: ~0.1 (tasa de aprendizaje)
- min_samples_split: ~10 (mínimo para dividir)
- min_samples_leaf: ~4 (mínimo en hojas)
- subsample: ~0.8 (fracción de datos por árbol)

Estos valores balancean complejidad y generalización.

**Analogía**: Es como afinar un instrumento musical: cada perilla (hiperparámetro) afecta el sonido final, y hay una combinación óptima que produce la mejor melodía.

**Mini-glosario**:

- **n_estimators**: Número de árboles en el ensemble
- **learning_rate**: Cuánto aprende cada árbol nuevo
- **max_depth**: Complejidad máxima de cada árbol

---

# CATEGORÍA 5: MÉTRICAS DE EVALUACIÓN

## Pregunta 20

**¿Por qué usaste ROC-AUC como métrica principal en lugar de Accuracy?**

**Respuesta**: Accuracy es engañosa con clases desbalanceadas: un modelo que predice siempre "No Churn" tendría 73% accuracy pero sería inútil. ROC-AUC mide la capacidad del modelo para distinguir entre clases en todos los umbrales de decisión, siendo más robusta al desbalance. Un ROC-AUC de 0.8503 significa que hay 85% de probabilidad de que el modelo rankee correctamente un caso positivo sobre uno negativo.

**Analogía**: Es como evaluar un detector de metales en un aeropuerto. No importa cuántas personas "normales" deja pasar (accuracy); importa que detecte las armas cuando las hay (ROC-AUC).

**Mini-glosario**:

- **ROC-AUC**: Área bajo la curva ROC (0.5 = aleatorio, 1.0 = perfecto)
- **Umbral de decisión**: Punto de corte para clasificar positivo/negativo
- **True Positive Rate**: Proporción de positivos correctamente identificados

---

## Pregunta 21

**¿Qué es la matriz de confusión y cómo la interpretas?**

**Respuesta**: La matriz de confusión muestra los 4 resultados posibles:

- **TN (True Negative)**: Predijo No Churn, era No Churn
- **FP (False Positive)**: Predijo Churn, era No Churn
- **FN (False Negative)**: Predijo No Churn, era Churn (el más costoso)
- **TP (True Positive)**: Predijo Churn, era Churn

En el modelo: TN=~850, FP=~180, FN=~75, TP=~300. Los FN son críticos porque son clientes que abandonarán sin que los detectemos.

**Analogía**: Es como un diagnóstico médico: TN = sano diagnosticado sano, FP = sano diagnosticado enfermo (susto innecesario), FN = enfermo diagnosticado sano (peligroso), TP = enfermo diagnosticado enfermo.

**Mini-glosario**:

- **Verdadero/Falso**: Si la predicción fue correcta o no
- **Positivo/Negativo**: La clase predicha
- **Matriz de confusión**: Tabla 2x2 de resultados

---

## Pregunta 22

**¿Cuál es la diferencia entre Precision y Recall, y cuál es más importante en este problema?**

**Respuesta**:

- **Precision**: De los que predije como Churn, ¿cuántos realmente lo son? (TP/(TP+FP)) = ~52%
- **Recall**: De los que realmente son Churn, ¿cuántos detecté? (TP/(TP+FN)) = ~80%

En churn prediction, **Recall es más importante** porque el costo de no detectar un cliente que abandonará (FN) es mayor que el costo de contactar a uno que no iba a abandonar (FP). Preferimos "molestar" a algunos clientes leales que perder clientes en riesgo.

**Analogía**: En un detector de incendios, preferimos falsas alarmas (bajo precision) a no detectar un incendio real (bajo recall). El costo de un incendio no detectado es catastrófico.

**Mini-glosario**:

- **Precision**: Exactitud de las predicciones positivas
- **Recall (Sensibilidad)**: Cobertura de los casos positivos reales
- **F1-Score**: Media armónica de Precision y Recall

---

## Pregunta 23

**¿Qué es el F1-Score y cuándo es útil?**

**Respuesta**: El F1-Score es la media armónica de Precision y Recall: F1 = 2 × (Precision × Recall) / (Precision + Recall). En el modelo, F1 = ~0.63. Es útil cuando necesitas un balance entre ambas métricas y cuando las clases están desbalanceadas. La media armónica penaliza valores extremos, así que un F1 alto requiere que ambas métricas sean razonablemente buenas.

**Analogía**: Es como la nota final de un curso que requiere aprobar tanto teoría como práctica. No puedes compensar un 0 en práctica con un 10 en teoría; necesitas un balance.

**Mini-glosario**:

- **Media armónica**: Tipo de promedio que penaliza valores bajos
- **Trade-off**: Compromiso entre dos objetivos opuestos
- **Balance**: Equilibrio entre Precision y Recall

---

### Pregunta 24

**¿Cómo validaste la robustez del modelo antes de producción?**

**Respuesta**: Se realizó validación de robustez con 5 semillas diferentes (42, 123, 456, 789, 2024), entrenando y evaluando el modelo con cada una. Los criterios de aceptación fueron:

1. Desviación estándar de ROC-AUC $< 0.02$ (obtenido: 0.0065)
2. Rango de variación $< 0.05$ (obtenido: 0.0163)
3. ROC-AUC promedio $> 0.80$ (obtenido: 0.8513)

El modelo pasó todos los criterios, indicando estabilidad para producción.

**Analogía**: Es como probar un carro en diferentes condiciones (lluvia, sol, noche, día) antes de venderlo. Si funciona bien en todas, puedes confiar en que funcionará para el cliente.

**Mini-glosario**:

- **Robustez**: Estabilidad del modelo ante variaciones
- **Semilla aleatoria**: Valor que controla la aleatoriedad
- **Criterios de aceptación**: Umbrales mínimos para producción

---

## CATEGORÍA 6: INTERPRETABILIDAD Y FEATURE IMPORTANCE

### Pregunta 25

**¿Cuáles son las 5 variables más importantes para predecir churn?**

**Respuesta**: Las 5 variables más importantes según el modelo son:

1. **Contract_Month-to-month** (~25%): Contratos mensuales tienen alto riesgo
2. **tenure** (~18%): Antigüedad del cliente
3. **TotalCharges** (~12%): Monto total pagado
4. **MonthlyCharges** (~10%): Cargo mensual
5. **InternetService_Fiber optic** (~8%): Servicio de fibra óptica

Estas variables explican más del 70% de la capacidad predictiva del modelo.

**Analogía**: Es como identificar los factores principales de éxito académico: horas de estudio, asistencia a clase, calidad del sueño. Algunos factores pesan más que otros.

**Mini-glosario**:

- **Feature Importance**: Peso de cada variable en las predicciones
- **Importancia relativa**: Porcentaje de contribución de cada variable
- **Variables predictoras**: Características usadas para predecir

---

## Pregunta 26

**¿Por qué el tipo de contrato es la variable más importante?**

**Respuesta**: Los contratos mes a mes tienen 42% de churn vs 11% en contratos anuales y 3% en bianuales. Esto se debe a que: (1) no hay penalización por cancelar, (2) indica menor compromiso inicial del cliente, (3) facilita la comparación con competidores. El modelo aprende que este es el predictor más fuerte de abandono.

**Analogía**: Es como la diferencia entre alquilar y comprar una casa. El inquilino (mes a mes) puede irse fácilmente; el propietario (contrato largo) tiene más razones para quedarse.

**Mini-glosario**:

- **Contrato mes a mes**: Sin compromiso a largo plazo
- **Lock-in**: Efecto de retención por compromiso contractual
- **Costo de cambio**: Barreras para cambiar de proveedor

---

## Pregunta 27

**¿Cómo explicarías las predicciones del modelo a un ejecutivo no técnico?**

**Respuesta**: El modelo analiza el perfil de cada cliente y asigna una probabilidad de abandono (0-100%). Los factores principales son: tipo de contrato, antigüedad, servicios contratados y montos pagados. Por ejemplo: "Este cliente tiene 85% de probabilidad de abandonar porque tiene contrato mensual, solo 3 meses de antigüedad, y no tiene servicios de seguridad online. Recomendamos ofrecerle un descuento por contrato anual."

**Analogía**: Es como un semáforo de riesgo: verde (bajo riesgo), amarillo (atención), rojo (acción urgente). El modelo nos dice el color y por qué.

**Mini-glosario**:

- **Probabilidad de churn**: Riesgo estimado de abandono (0-1)
- **Scoring**: Asignación de puntuación de riesgo
- **Interpretabilidad**: Capacidad de explicar las predicciones

---

# CATEGORÍA 7: IMPACTO DE NEGOCIO Y ROI

## Pregunta 28

**¿Cómo calculaste el ROI del modelo de predicción de churn?**

**Respuesta**: Se usó la función `reporte_negocio()` con parámetros:

- LTV (Lifetime Value) por cliente: $2,000
- Costo de campaña de retención: $150
- Tasa de éxito de retención: 50%

ROI = (Clientes retenidos × LTV - Costo total de campañas) / Costo total × 100

Con ~300 TP y 50% de éxito, se retienen ~150 clientes, generando $300,000 en valor vs ~$72,000 en costos = ROI de ~316%.

**Analogía**: Es como invertir en publicidad: gastas $1 y recuperas $4. El modelo te dice dónde invertir ese dólar para máximo retorno.

**Mini-glosario**:

- **ROI**: Return on Investment (retorno sobre inversión)
- **LTV**: Valor total del cliente durante su vida útil
- **Costo de adquisición**: Gasto para conseguir un nuevo cliente

---

## Pregunta 29

**¿Cuál es el costo de un Falso Negativo vs un Falso Positivo en este contexto?**

**Respuesta**:

- **Falso Negativo (FN)**: No detectar un cliente que abandonará. Costo = LTV perdido = $2,000
- **Falso Positivo (FP)**: Contactar a un cliente que no iba a abandonar. Costo = Costo de campaña = $150

El FN es ~13 veces más costoso que el FP. Por eso optimizamos para alto Recall aunque sacrifiquemos algo de Precision.

**Analogía**: En medicina, no detectar un cáncer (FN) es mucho peor que hacer una biopsia innecesaria (FP). El costo del error no es simétrico.

**Mini-glosario**:

- **Costo asimétrico**: Cuando los errores tienen diferentes consecuencias
- **Matriz de costos**: Asignación de costos a cada tipo de error
- **Optimización por costo**: Minimizar el costo total de errores

---

## Pregunta 30

**¿Qué estrategias de retención recomendarías basándote en los insights del modelo?**

**Respuesta**: Basándome en las variables importantes:

1. **Migración a contratos largos**: Ofrecer descuentos por contratos anuales/bianuales
2. **Programa de onboarding**: Atención especial en primeros 12 meses
3. **Bundles de servicios**: Promover OnlineSecurity (seguridad en línea), TechSupport (soporte técnico) - reducen churn
4. **Revisión de precios**: Analizar clientes con cargos altos vs valor percibido
5. **Retención proactiva**: Contactar clientes con probabilidad >70% antes de que decidan irse

**Analogía**: Es como un programa de fidelización de aerolínea: millas, upgrades, atención preferencial. Cada beneficio ataca un factor de riesgo específico.

**Mini-glosario**:

- **Estrategia de retención**: Acciones para evitar el abandono
- **Onboarding**: Proceso de integración de nuevos clientes
- **Bundle**: Paquete de servicios combinados

---

# CATEGORÍA 8: ASPECTOS TÉCNICOS AVANZADOS

## Pregunta 31

**¿Qué es el overfitting y cómo lo previniste?**

**Respuesta**: Overfitting ocurre cuando el modelo memoriza los datos de entrenamiento pero falla con datos nuevos. Se previno mediante:

1. **Validación cruzada**: Evaluar en múltiples particiones
2. **Regularización**: Parámetros como min_samples_leaf, max_depth
3. **Early stopping implícito**: Limitar n_estimators
4. **Validación de robustez**: Probar con múltiples semillas
5. **Comparar train vs test score**: Diferencia pequeña indica buen ajuste

El modelo tiene CV score ~0.84 y test score ~0.85, indicando buena generalización.

**Analogía**: Es como un estudiante que memoriza respuestas vs uno que entiende conceptos. El primero falla con preguntas nuevas; el segundo puede resolver problemas que nunca vio.

**Mini-glosario**:

- **Overfitting**: Sobreajuste a los datos de entrenamiento
- **Underfitting**: Modelo demasiado simple
- **Generalización**: Rendimiento en datos no vistos

---

## Pregunta 32

**¿Por qué usaste random_state=42 en todo el proyecto?**

**Respuesta**: random_state=42 es una semilla que controla la aleatoriedad, garantizando reproducibilidad. Esto significa que cualquier persona que ejecute el código obtendrá exactamente los mismos resultados. Es esencial para: (1) debugging, (2) comparación justa de modelos, (3) documentación científica, (4) auditoría. El 42 es una convención popular (referencia a "The Hitchhiker's Guide to the Galaxy").

**Analogía**: Es como usar la misma receta con las mismas medidas exactas. Si cambias algo, el resultado puede variar, pero con la misma receta siempre obtienes el mismo plato.

**Mini-glosario**:

- **Reproducibilidad**: Capacidad de obtener los mismos resultados

- **Semilla aleatoria**: Valor inicial para generador de números aleatorios
- **Determinismo**: Mismo input produce mismo output

---

## Pregunta 33

**¿Cómo guardarías el modelo para usarlo en producción?**

**Respuesta**: Se usó `joblib.dump()` para guardar:

1. **churn_model.pkl**: El modelo entrenado (~0.5 MB)
2. **preprocessor.pkl**: El pipeline de preprocesamiento (~0.1 MB)
3. **metadata.json**: Información del modelo, métricas, versiones

Para predecir en producción: cargar ambos archivos, aplicar preprocessor.transform() a los datos nuevos, y luego model.predict_proba(). El tamaño total (~0.6 MB) es adecuado para deployment en Render/Railway.

**Analogía**: Es como guardar una receta completa: los ingredientes (preprocessor), las instrucciones (modelo), y las notas del chef (metadata). Cualquiera puede reproducir el plato.

**Mini-glosario**:

- **Serialización**: Convertir objeto a archivo
- **Pickle/Joblib**: Formatos para guardar objetos Python
- **Deployment**: Poner el modelo en producción

---

## Pregunta 34

**¿Qué consideraciones tendrías para monitorear el modelo en producción?**

**Respuesta**: Monitoreo esencial incluye:

1. **Data drift**: ¿Los datos nuevos son similares a los de entrenamiento?
2. **Concept drift**: ¿La relación entre variables y churn cambió?
3. **Métricas en vivo**: Comparar predicciones vs resultados reales
4. **Latencia**: Tiempo de respuesta de las predicciones
5. **Volumen**: Número de predicciones por período
6. **Alertas**: Notificar si métricas caen bajo umbrales

Recomendación: reentrenar mensualmente o cuando las métricas degraden >5%.

**Analogía**: Es como el mantenimiento de un carro: revisiones periódicas, alertas del tablero, y reparaciones cuando algo falla. No esperas a que se descomponga.

**Mini-glosario**:

- **Data drift**: Cambio en la distribución de los datos
- **Concept drift**: Cambio en la relación datos-resultado
- **MLOps**: Operaciones de Machine Learning en producción

---

# CATEGORÍA 9: PREGUNTAS DE SÍNTESIS Y REFLEXIÓN

## Pregunta 35

**Si tuvieras que explicar todo el proyecto en 2 minutos, ¿qué dirías?**

**Respuesta**: "Desarrollamos un modelo de Machine Learning para predecir qué clientes de telecomunicaciones abandonarán el servicio. Analizamos 7,043 clientes con 21 características. El principal desafío fue el desbalance de clases (27% churn), que resolvimos con Undersampling. Evaluamos 7 algoritmos y Logistic Regression Optimizado fue el mejor con 85% ROC-AUC y 79% Recall. Las variables más importantes son tipo de contrato, antigüedad y cargos. El modelo puede generar ROI de 300%+ al permitir campañas de retención focalizadas. Está validado, es robusto, y listo para producción."

**Analogía**: Es como el elevator pitch de una startup: problema, solución, resultados, en el tiempo que dura un viaje en ascensor.

**Mini-glosario**:

- **Elevator pitch**: Presentación concisa de un proyecto
- **Key takeaways**: Puntos principales a recordar
- **Executive summary**: Resumen para tomadores de decisiones

---

## Pregunta 36

**¿Cuáles fueron los principales desafíos técnicos y cómo los resolviste?**

**Respuesta**: Los principales desafíos fueron:

1. **Desbalance de clases**: Resuelto con Undersampling (seleccionado tras comparar SMOTE, SMOTE+Tomek y Undersampling)
2. **Valores faltantes**: Imputación lógica basada en contexto (tenure=0)
3. **Variables mixtas**: Pipeline con ColumnTransformer para diferentes tipos
4. **Selección de modelo**: Evaluación sistemática de 7 algoritmos
5. **Optimización**: RandomizedSearchCV para hiperparámetros
6. **Validación**: Múltiples semillas para garantizar robustez

Cada desafío se abordó con metodología rigurosa y documentación.

**Analogía**: Es como escalar una montaña: cada obstáculo (grieta, tormenta, altitud) requiere una técnica específica. La preparación y el equipo adecuado son clave.

**Mini-glosario**:

- **Desafío técnico**: Problema que requiere solución especializada
- **Metodología**: Proceso sistemático para resolver problemas
- **Best practices**: Mejores prácticas de la industria

---

## Pregunta 37

**¿Qué harías diferente si empezaras el proyecto de nuevo?**

**Respuesta**: Mejoras potenciales:

1. **Más Feature Engineering**: Crear variables de tendencia temporal
2. **Análisis de cohortes**: Segmentar por fecha de adquisición
3. **Modelos de supervivencia**: Predecir cuándo abandonará, no solo si
4. **Explicabilidad**: Implementar SHAP values para interpretación local
5. **A/B testing**: Diseñar experimento para validar impacto real
6. **Pipeline automatizado**: MLflow para tracking de experimentos

El proyecto actual es sólido, pero siempre hay espacio para mejora.

**Analogía**: Es como renovar una casa: la estructura es buena, pero podrías mejorar la cocina, agregar un baño, o modernizar la electricidad.

**Mini-glosario**:

- **Análisis de cohortes**: Estudiar grupos por período de ingreso
- **SHAP values**: Explicación de predicciones individuales
- **MLflow**: Herramienta para gestión de experimentos ML

---

## Pregunta 38

**¿Cómo manejarías datos nuevos que tienen categorías que no existían en el entrenamiento?**

**Respuesta**: Este es el problema de "categorías no vistas". Soluciones:

1. **handle_unknown='ignore'** en OneHotEncoder: Ignora categorías nuevas
2. **Categoría "Otros"**: Agrupar categorías raras durante entrenamiento
3. **Reentrenamiento**: Incluir nuevas categorías periódicamente
4. **Validación de entrada**: Alertar cuando aparecen valores inesperados
5. **Fallback**: Usar predicción por defecto si hay error

En el proyecto se usó handle_unknown='ignore' para robustez.

**Analogía**: Es como un traductor que encuentra una palabra nueva: puede ignorarla, usar una aproximación, o pedir ayuda. Lo importante es no fallar completamente.

**Mini-glosario**:

- **Categoría no vista**: Valor que no existía en entrenamiento
- **Handle unknown**: Estrategia para manejar valores nuevos
- **Robustez de entrada**: Tolerancia a datos inesperados

---

## Pregunta 39

**¿Cómo comunicarías los resultados a diferentes audiencias?**

**Respuesta**: Adaptación por audiencia:

- **Ejecutivos**: ROI, impacto en ingresos, recomendaciones de acción
- **Marketing**: Segmentos de riesgo, variables accionables, campañas sugeridas
- **Técnicos**: Métricas detalladas, arquitectura, código, reproducibilidad
- **Operaciones**: Cómo usar el modelo, qué datos necesita, frecuencia de actualización

Cada audiencia tiene diferentes necesidades y nivel de detalle.

**Analogía**: Es como explicar el clima: a un piloto le das datos técnicos, a un turista le dices "lleva paraguas", a un agricultor le hablas de precipitación acumulada.

**Mini-glosario**:

- **Stakeholder**: Persona interesada en el proyecto
- **Comunicación técnica**: Adaptada a expertos
- **Comunicación ejecutiva**: Enfocada en impacto y decisiones

---

## Pregunta 40

**¿Cuál es el siguiente paso después de este proyecto?**

**Respuesta**: Roadmap sugerido:

1. **Corto plazo (1-2 semanas)**: Desarrollar API REST con Flask/FastAPI
2. **Medio plazo (1 mes)**: Dashboard interactivo con Streamlit
3. **Producción (2 meses)**: Deploy en cloud (Render/Railway/AWS)
4. **Integración (3 meses)**: Conectar con CRM de la empresa
5. **Automatización (6 meses)**: Pipeline de reentrenamiento automático
6. **Expansión**: Modelos de upselling, cross-selling, lifetime value

El modelo actual es el MVP; el valor real viene de la implementación y uso continuo.

**Analogía**: Es como construir una casa: el proyecto arquitectónico (modelo) está listo, ahora viene la construcción (deployment), la mudanza (integración), y el mantenimiento (monitoreo).

**Mini-glosario**:

- **MVP**: Minimum Viable Product (producto mínimo viable)
- **Roadmap**: Plan de desarrollo a futuro
- **API REST**: Interfaz para consumir el modelo programáticamente

---

# GLOSARIO GENERAL

| Término | Definición |
|---|---|
| **Churn** | Abandono de clientes |
| **ROC-AUC** | Métrica de capacidad discriminativa (0.5-1.0) |
| **Undersampling** | Técnica de reducción de clase mayoritaria |
| **SMOTE** | Técnica de oversampling sintético |
| **Logistic Regression** | Modelo lineal de clasificación, interpretable |
| **Feature Engineering** | Creación de nuevas variables |
| **Cross-validation** | Validación con múltiples particiones |
| **Precision** | Exactitud de predicciones positivas |
| **Recall** | Cobertura de casos positivos reales |
| **Overfitting** | Sobreajuste a datos de entrenamiento |
| **Pipeline** | Secuencia de transformaciones |
| **Hiperparámetro** | Configuración externa del modelo |
| **LTV** | Lifetime Value (valor del cliente) |
| **ROI** | Return on Investment |
| **Data drift** | Cambio en distribución de datos |
| **Deployment** | Puesta en producción |

---

*Documento generado para la sustentación del proyecto de predicción de Customer Churn Bootcamp de IA - Nivel Básico*