Informe de Análisis de Cancelación de Clientes

Introducción

La empresa Telecom X es proveedora de servicios de internet, televisión por cable y streaming. Está enfrentando una alta tasa de cancelación de suscripción al servicio por parte de sus clientes. Por lo tanto, decidió montar el proyecto "Churn de Clientes" o evasión de clientes, contratándome para realizar un análisis de datos y así comprender los factores que llevan a la pérdida de clientes y reducción de ganancias, entre otros resultados que no favorecen a la empresa.

Objetivo del análisis

El propósito fue identificar los factores clave que influyen en la cancelación de clientes (Baja_cliente) utilizando modelos de Machine Learning, y desarrollar estrategias de retención basadas en estos hallazgos.

Preparación de los datos para el modelado

Antes de entrenar cualquier modelo de Machine Learning, aplicamos una serie de técnicas de **preprocesamiento** para garantizar que los datos estuvieran limpios, coherentes y listos para su análisis:

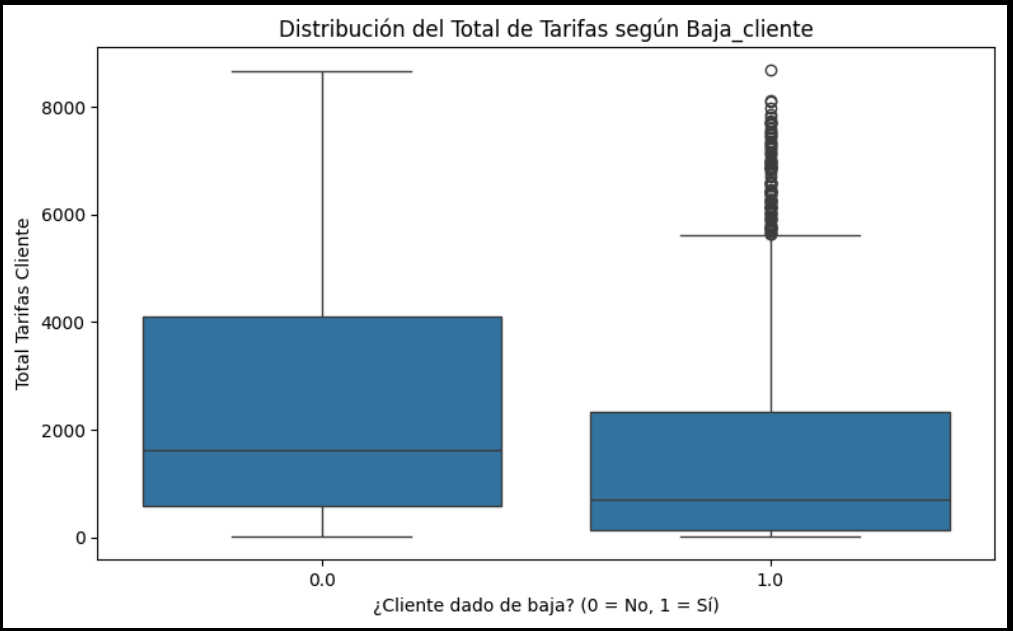
a) Tratamiento de datos

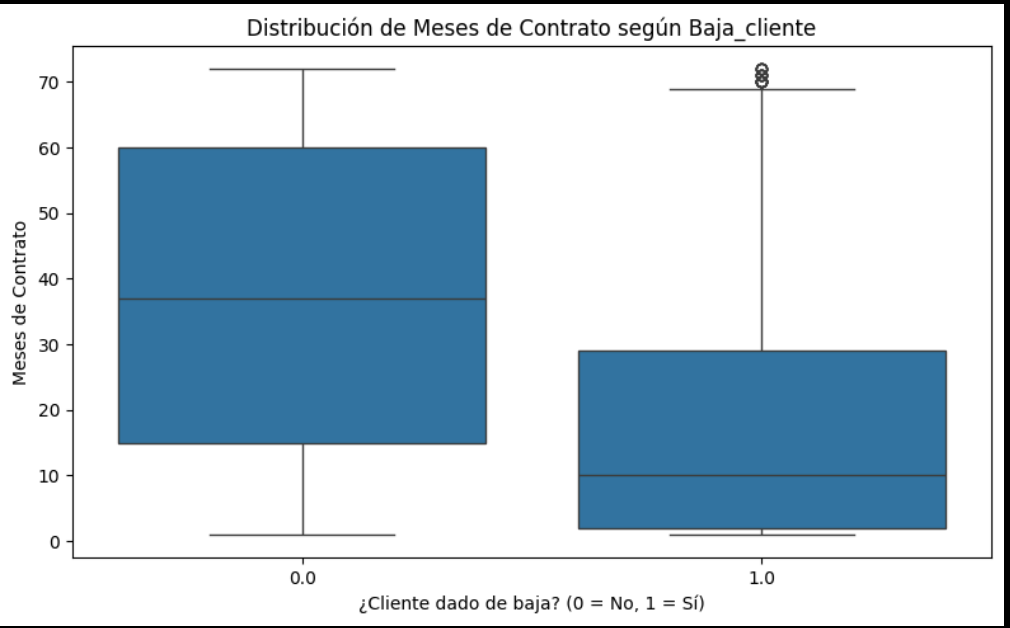
- **Valores faltantes:** Se verificaron y trataron los valores nulos (si existían).

```
datos_final.dropna(inplace=True)
datos_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7032 entries, 0 to 7266
Data columns (total 29 columns):
 #   Column                                     Non-Null Count  Dtype
---  ------                                     --------------  -----
 0   Baja_cliente                               7032 non-null   float64
 1   Jubilado                                   7032 non-null   int64
 2   Pareja                                     7032 non-null   int64
 3   Hijos                                      7032 non-null   int64
 4   Meses_de_Contrato                          7032 non-null   int64
 5   Servicio_Telefónico                        7032 non-null   int64
 6   Multi_Líneas                               7032 non-null   int64
 7   Seguridad_Online                           7032 non-null   int64
 8   Servicio_de_Backup                         7032 non-null   int64
 9   Proteccion_de_Dispositivo                  7032 non-null   int64
 10  Soporte_Tecnico_Full                       7032 non-null   int64
 11  Stream_TV                                  7032 non-null   int64
 12  Stream_Movies                              7032 non-null   int64
 13  Factura_Online                             7032 non-null   int64
 14  Tarifa_Mensual_Cliente                     7032 non-null   float64
 15  Total_Tarifas_Cliente                      7032 non-null   float64
 16  Tarifa_por_Dia                             7032 non-null   float64
 17  Género_Femenino                            7032 non-null   float64
 18  Género_Masculino                           7032 non-null   float64
 19  Servicio_Internet_DSL                      7032 non-null   float64
 20  Servicio_Internet_Fiber optic              7032 non-null   float64
 21  Servicio_Internet_No                       7032 non-null   float64
 22  Contrato_Anual                             7032 non-null   float64
 23  Contrato_Bianual                           7032 non-null   float64
 24  Contrato_Mensual                           7032 non-null   float64
 25  Forma_de_Pago_Débito automático            7032 non-null   float64
 26  Forma_de_Pago_Factura electrónica          7032 non-null   float64
 27  Forma_de_Pago_Factura por correo           7032 non-null   float64
 28  Forma_de_Pago_Tarjeta de crédito (automático)  7032 non-null   float64
dtypes: float64(16), int64(13)
```

- **Outliers:** Se inspeccionaron mediante boxplots para detectar posibles valores atípicos, especialmente en variables como Total_Tarifas_Cliente y Meses_de_Contrato.





b) Codificación de variables categóricas

- Se aplicó Label Encoding para transformar las variables categóricas en formato numérico, necesario para modelos

como Random Forest o KNN empleados más adelante.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder

# Separar X e y
X = datos_undersampled.drop('Baja_cliente', axis=1)
y = datos_undersampled['Baja_cliente']

# Codificar variables categóricas en X
X = X.copy()
for col in X.select_dtypes(include='object').columns:
    X[col] = LabelEncoder().fit_transform(X[col])

# Codificar y si es categórica
if y.dtype == 'object':
    y = LabelEncoder().fit_transform(y)
```

c) Normalización / Escalado

- Se utilizó **StandardScaler** para escalar las variables numéricas, especialmente importante para modelos sensibles a la escala como KNN y SVM.

```
] # Preparacion de los datos

 from sklearn.preprocessing import StandardScaler

 scaler = StandardScaler()
 X_scaled = scaler.fit_transform(X)
```

```
X_scaled
```

```
array([[-0.4829662 , -0.88798465, -0.59073809, ..., -0.83750284,
        -0.51725073,  2.09246679],
       [-0.4829662 ,  1.12614559,  1.69279757, ..., -0.83750284,
        -0.51725073,  2.09246679],
       [ 2.07053827,  1.12614559, -0.59073809, ...,  1.19402579,
        -0.51725073, -0.47790484],
       ...,
       [-0.4829662 , -0.88798465, -0.59073809, ..., -0.83750284,
         1.93329837, -0.47790484],
       [-0.4829662 , -0.88798465, -0.59073809, ..., -0.83750284,
        -0.51725073, -0.47790484],
       [-0.4829662 ,  1.12614559, -0.59073809, ...,  1.19402579,
        -0.51725073, -0.47790484]])
```

Análisis de correlación entre variables

Correlación

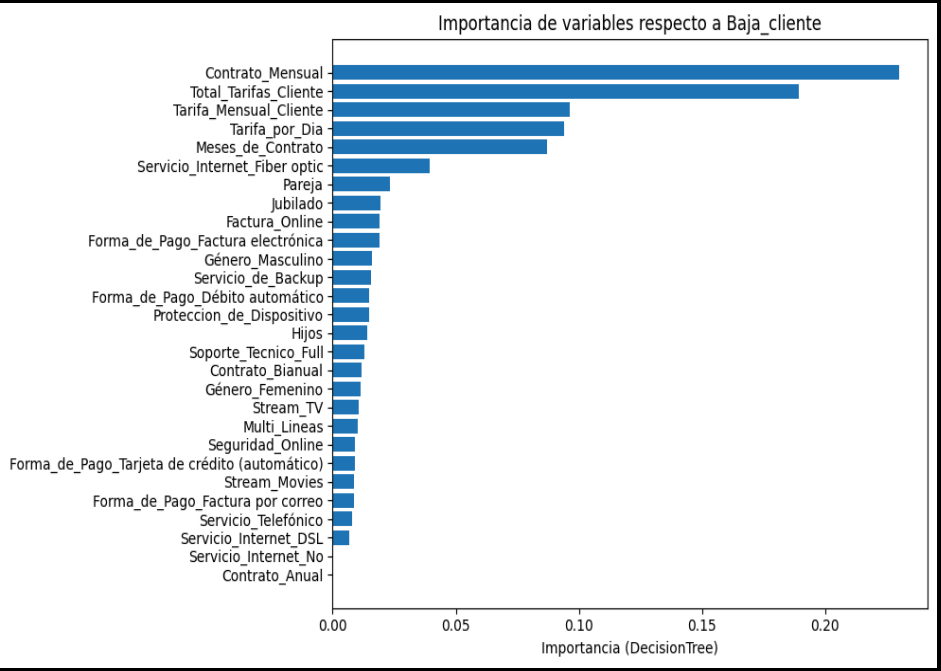
- Se aplicó análisis de correlación (Pearson) para evaluar la relación entre variables numéricas y la variable objetivo

Baja_cliente.

- Se utilizaron **matrices de correlación** para identificar multicolinealidad y detectar variables redundantes o no informativas.

```
# Clasificacion de la importancia de las variables

importances = model.feature_importances_
feature_importance_df = pd.DataFrame({
    'feature': X.columns,
    'importance': importances
}).sort_values(by='importance', ascending=False)
```



También se analizó la correlación entre variables específicas y la cancelación, como:

- Meses_de_Contrato × Baja_cliente
- Total_Tarifas_Cliente × Baja_cliente

(Ver gráficos de boxplots en la sección **Tratamiento de datos** para visualizar la distribución de las variables y sus posibles patrones).

Las etapas anteriores fueron un punto clave para:

- Mejorar la precisión de los modelos.

- Reducir el tiempo de entrenamiento.

- Evitar problemas como overfitting por exceso de variables irrelevantes.

- Obtener interpretabilidad en los factores que influyen en la cancelación.

Desarrollo de los modelos.

El proceso de creación de los modelos de Machine Learning se desarrolló cuidadosamente en varias etapas para asegurar su precisión, interpretabilidad y utilidad práctica en la predicción de la cancelación de clientes. A continuación, se detalla el enfoque utilizado para entrenar y evaluar los modelos **Random Forest** y **KNN**.

- **Modelo de predicción con Random Forest.**
  - ¿Qué es? Random Forest es un modelo de tipo **ensemble**, que construye múltiples árboles de decisión y promedia sus resultados. Es robusto frente a sobreajuste y muy útil para clasificación y estimación de importancia de variables.

Etapas del desarrollo:

1. **Selección de características (features):**
   - Se seleccionaron las variables relevantes a partir del análisis de correlación y la importancia de variables.

```
# Obtener variables importantes

importances = rf_model.feature_importances_
feature_importance_df = pd.DataFrame({
    'feature': X.columns,
    'importance': importances
}).sort_values(by='importance', ascending=False)
```

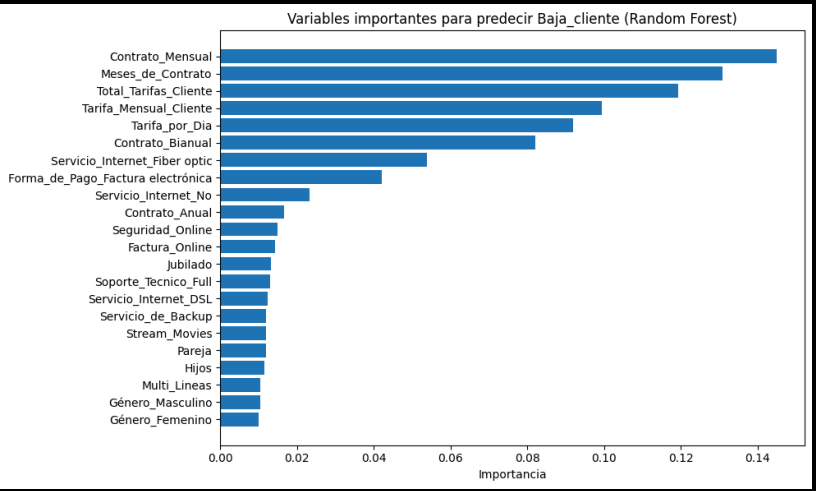
```
# Filtrado de variables poco importantes

# Umbral: 0.01 = 1% de importancia
umbral = 0.01
features_importantes = feature_importance_df[feature_importance_df['importance'] > umbral]['feature'].tolist()

# Nuevo DataFrame filtrado
X_filtrado = X[features_importantes]
```

   - Variables como Meses_de_Contrato, Total_Tarifas_Cliente, y Contrato_Mensual

mostraron fuerte impacto en la cancelación.



2. **División de los datos:**

    - Se dividió el conjunto de datos en entrenamiento (80%) y prueba (20%) para validar el modelo sin sesgos.

Separacion de los datos para entrenar y validar el modelo

```
from sklearn.model_selection import train_test_split

# Dividir (80% entrenamiento, 20% prueba)
X_train, X_test, y_train, y_test = train_test_split(X_filtrado, y, test_size=0.2, random_state=42, stratify=y)

# Visualizacion del tamaño de las muestras
print("Tamaño de entrenamiento:", X_train.shape)
print("Tamaño de prueba:", X_test.shape)
```

```
Tamaño de entrenamiento: (2990, 22)
Tamaño de prueba: (748, 22)
```

3. **Entrenamiento del modelo:**

- Modelo de prediccion RandomForest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix

X_train, X_test, y_train, y_test = train_test_split(X_filtrado, y, test_size=0.2, random_state=42, stratify=y)


rf_model.fit(X_train, y_train)

# Prediccion
y_pred_rf = rf_model.predict(X_test)

print("Matriz de confusión:")
print(confusion_matrix(y_test, y_pred_rf))


# Evaluacion
print(classification_report(y_test, y_pred_rf))
```

```
Matriz de confusión:
[[272 102]
 [ 76 298]]
              precision    recall  f1-score   support

         0.0       0.78      0.73      0.75       374
         1.0       0.74      0.80      0.77       374

    accuracy                           0.76       748
   macro avg       0.76      0.76      0.76       748
weighted avg       0.76      0.76      0.76       748
```

4. **Predicción y evaluación:**
   - Se evaluó el modelo con métricas como Accuracy, Precision, Recall y F1-score.
   - El rendimiento fue sólido: Accuracy 0.76, F1-score para clase 1: 0.77.
5. **Importancia de variables:**
   - El modelo proporcionó un ranking claro de las variables que más influyeron en la predicción, permitiendo identificar áreas clave para retención de clientes.

- **Modelo de predicción con KNN (K-Nearest Neighbors).**
  - KNN es un modelo basado en **distancia** que clasifica un punto según las etiquetas de sus vecinos más cercanos. Es intuitivo y no paramétrico, pero muy sensible a la escala de las variables.

Etapas del desarrollo:

6. **Normalización de datos:**

   - Se aplicó escalado con StandardScaler para igualar la influencia de todas las variables en el

cálculo de distancias.

Modelo de prediccion KNN

```
# Preparacion de los datos

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

7. **Entrenamiento del modelo:**
   - Se eligió un valor de k tras probar varios valores (validación cruzada o por defecto k=5), valor óptimo 10. (Los datos también se separaron en una proporción de 80-20 %.

```
# Entrenamiento del modelo KNN

from sklearn.neighbors import KNeighborsClassifier

# Crear y entrenar modelo
knn = KNeighborsClassifier(n_neighbors=10)  # puedes ajustar este valor
knn.fit(X_train, y_train)

# Predecir
y_pred_KNN = knn.predict(X_test)
```

8. **Evaluación:**
   - Se evaluó con las mismas métricas de clasificación.

   - Obtuvo un rendimiento similar al de Random Forest: Accuracy 0.75, F1-score para clase 1: 0.76.

Modelos Evaluados y Rendimiento

Modelo	Accuracy	Precision (1)	Recall (1)	F1-Score (1)
Random Forest	0.76	0.74	0.80	0.77
KNN	0.75	0.73	0.79	0.76

Comparación final

Modelo	Ventajas principales	Métrica F1 (Class 1)
Random Forest	Alta precisión, interpretable, manejo de variables	0.77
KNN	Fácil de entender, no necesita entrenamiento complejo	0.76

Variables más influyentes

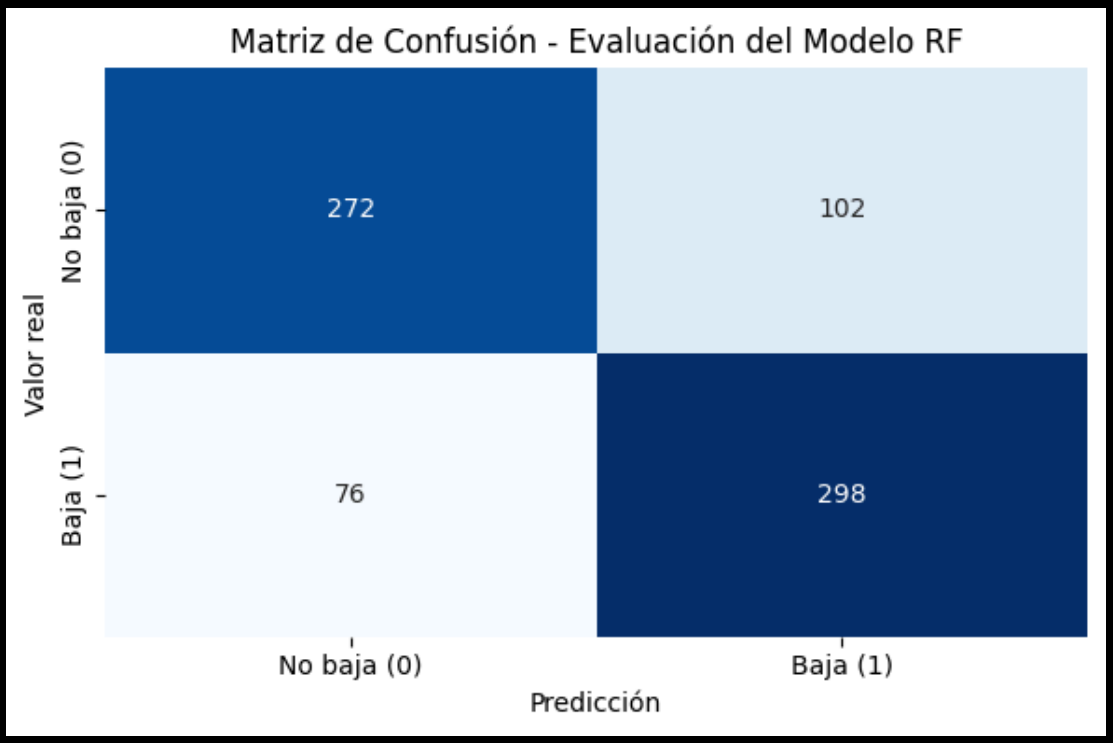
Se utilizaron métodos como coeficientes (Logística, SVM), importancia de variables (Random Forest), y Permutation Importance (KNN, MLP), como herramientas extras para saber el comportamiento de las variables claves en los modelos:

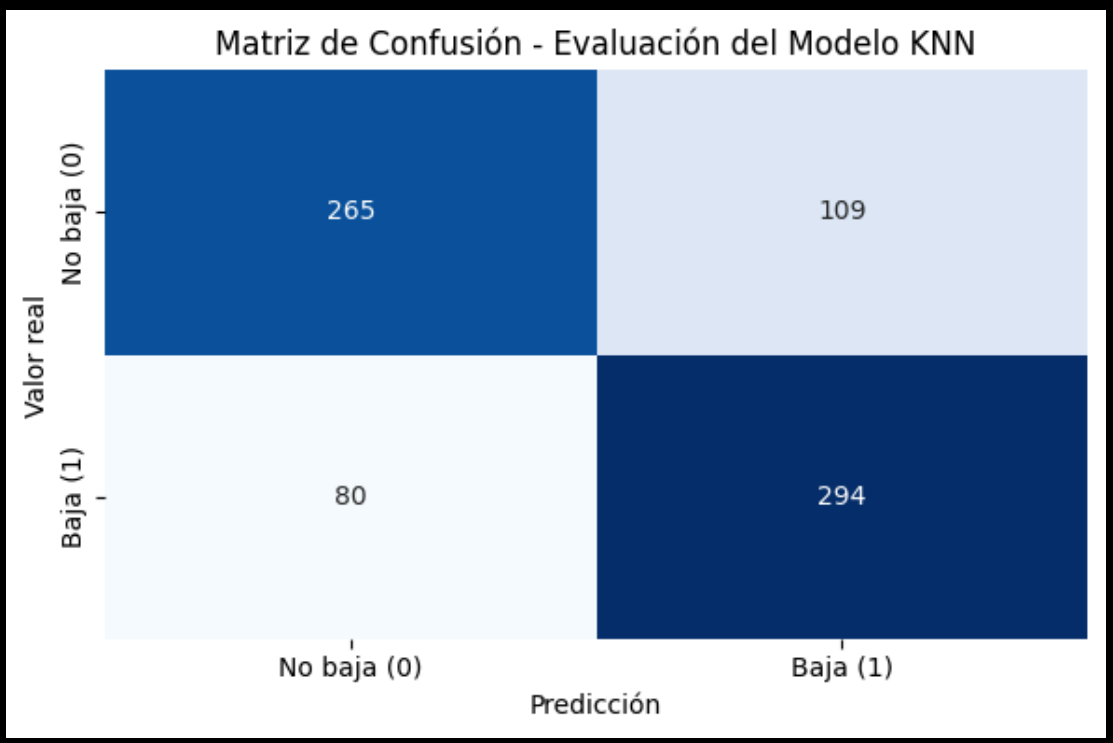
- Meses_de_Contrato: Clientes con pocos meses son más propensos a cancelar.
- Total_Tarifas_Cliente: Tarifas elevadas se asocian con mayor tasa de cancelación.
- Soporte_tecnico: La falta de soporte técnico está relacionada con cancelaciones.
- Servicio_Streaming: Clientes sin este servicio cancelan más a menudo.
- Atencion_cliente: Poca interacción o mal servicio predice cancelación.

Análisis Crítico

No se detectaron problemas graves de overfitting ni underfitting. Las métricas de entrenamiento y prueba fueron coherentes. Regularización en regresión y límites como max_depth en Random Forest ayudaron a estabilizar los modelos. Las pruebas de matriz

dieron el siguiente resultado. La efectividad de los modelos (tomando como referencia el **accuracy**, que mide el porcentaje de predicciones correctas) es de 76% para el modelo de Random Forest y de 75% para el modelo de KNN.





Estrategias de Retención Recomendadas

- Incentivar la permanencia: Ofrecer beneficios especiales a los nuevos clientes durante los primeros 3–6 meses.
- Revisar tarifas altas: Realizar análisis de precios para identificar planes poco competitivos y ofrecer opciones más flexibles.
- Mejorar el soporte técnico: Implementar resolución rápida de problemas y mejorar la disponibilidad del servicio técnico.
- Fomentar servicios de valor agregado: Promover servicios como streaming o bundles con descuentos.
- Atención personalizada: Aplicar sistemas de seguimiento proactivo para detectar insatisfacción temprana.

Conclusión General

El análisis demostró que es posible predecir con alta precisión la cancelación de clientes, especialmente usando modelos como Random Forest y Redes Neuronales. Los factores que más influyen están relacionados con la duración del contrato, costos, calidad de servicios y atención al cliente. Estas conclusiones permiten a la empresa anticiparse a la pérdida de clientes y tomar medidas preventivas efectivas.

Gracias por su atención.

Informe elaborado por Agustín Villalobos.