Memoria Técnica

Predicción de la Intención de Compra en un Comercio-Online

THE BRIDGE DIGITAL TALENT ACCELERATOR

Bootcamp Data Science Online, Noviembre 2024

Luis Tamayo Ortiz

Indice

1 Introducción

El comercio electrónico ha transformado la forma en que los consumidores interactúan con las marcas, permitiéndoles acceder a productos y servicios desde cualquier lugar y en cualquier momento. En este entorno altamente competitivo, **las campañas de marketing personalizadas** se han convertido en una herramienta crucial para las empresas que buscan destacarse, ya que no solo mejoran la experiencia del cliente, sino que también generan un impacto directo en las tasas de conversión y la fidelización.

1.1 Importancia de la Personalización en el Marketing

Los consumidores actuales esperan experiencias personalizadas que se adapten a sus necesidades, preferencias y comportamientos de compra. Estudios demuestran que las campañas basadas en datos, como recomendaciones personalizadas o promociones dirigidas, pueden:
Incrementar la satisfacción del cliente al ofrecer productos y ofertas relevantes.
Aumentar las tasas de conversión al dirigirse a los usuarios correctos con mensajes específicos.
Construir relaciones a largo plazo, fortaleciendo la fidelidad hacia la marca.

1.2 Inversión Empresarial y Optimización

Las empresas destinan **miles de millones de dólares anuales** a campañas de marketing digital. Por ejemplo:
Se invierten en promedio entre un 10% y un 25% del presupuesto de ingresos en estrategias digitales, dependiendo del sector.
Los costos de adquisición de clientes (CAC) pueden dispararse si las campañas no son efectivas o no están dirigidas al público objetivo adecuado.

Una **selección correcta de los usuarios objetivo** puede:

- **Reducir Costes**: Al enfocar los esfuerzos en segmentos específicos, las empresas minimizan el gasto innecesario en audiencias que probablemente no conviertan.
- **Mejorar la Conversión**: Dirigir campañas hacia usuarios con alta probabilidad de compra maximiza el retorno de la inversión (ROI) en marketing.

1.3 Rol del Análisis de Datos

El análisis de datos, mediante técnicas de Machine Learning y modelos predictivos, permite a las empresas identificar patrones en el comportamiento del cliente. Esto no solo ayuda a predecir qué usuarios tienen más probabilidades de realizar una compra, sino que también facilita la segmentación, personalización y priorización de recursos, optimizando tanto la experiencia del usuario como los costos operativos.
En este contexto, construir modelos que permitan identificar y predecir la intención de compra es una estrategia esencial para maximizar la eficiencia del marketing y mejorar los resultados comerciales.

2 Caso particular de aplicación

Introducción al Caso de Estudio: La Tienda Online de Manuel

Manuel es el dueño de una tienda online que ofrece una amplia gama de productos a través de su plataforma digital. Consciente del creciente auge del comercio electrónico, ha identificado que su negocio tiene el potencial de competir en este mercado, pero enfrenta varios desafíos relacionados con las tasas de conversión y la efectividad de sus campañas de marketing.

2.1 Problemas Identificados

1. **Baja Tasa de Conversión**: Aunque su tienda recibe un volumen considerable de visitas diarias, el porcentaje de usuarios que realizan una compra es limitado, afectando los ingresos y el retorno de la inversión publicitaria.

2. **Altos Costos de Adquisición**: Manuel ha destinado recursos importantes a estrategias de marketing digital como campañas de anuncios y promociones generales, pero el rendimiento no ha sido el esperado debido a la falta de personalización y segmentación adecuada.

3. **Competencia Intensa**: En un mercado saturado, ofrecer productos relevantes y experiencias personalizadas se ha convertido en un requisito para captar y retener clientes.

2.2 La Oportunidad

Manuel comprende que optimizar sus estrategias de marketing y segmentar adecuadamente a sus usuarios puede marcar una gran diferencia. Identificar qué usuarios tienen una mayor intención de compra permitirá:

- Diseñar campañas más personalizadas y efectivas.
- Maximizar la conversión de visitas en ventas.
- Reducir significativamente los costos operativos y de adquisición de clientes.

2.3 El Rol de los Datos

La tienda de Manuel genera una gran cantidad de datos diarios, que incluyen información sobre las sesiones de los usuarios, el comportamiento de navegación, el tipo de visitante (nuevo o recurrente) y los productos visitados. Aprovechar este conjunto de datos para construir un modelo predictivo puede ayudar a Manuel a:

1. Identificar patrones en el comportamiento de los usuarios.

2. Predecir con mayor precisión quiénes son más propensos a realizar una compra.

3. Priorizar recursos en los segmentos más valiosos.

Este caso de estudio busca abordar los desafíos de Manuel mediante el desarrollo de un modelo de predicción de intención de compra, que le permitirá optimizar su estrategia comercial, aumentar sus ingresos y consolidar su posición en el mercado del comercio electrónico.

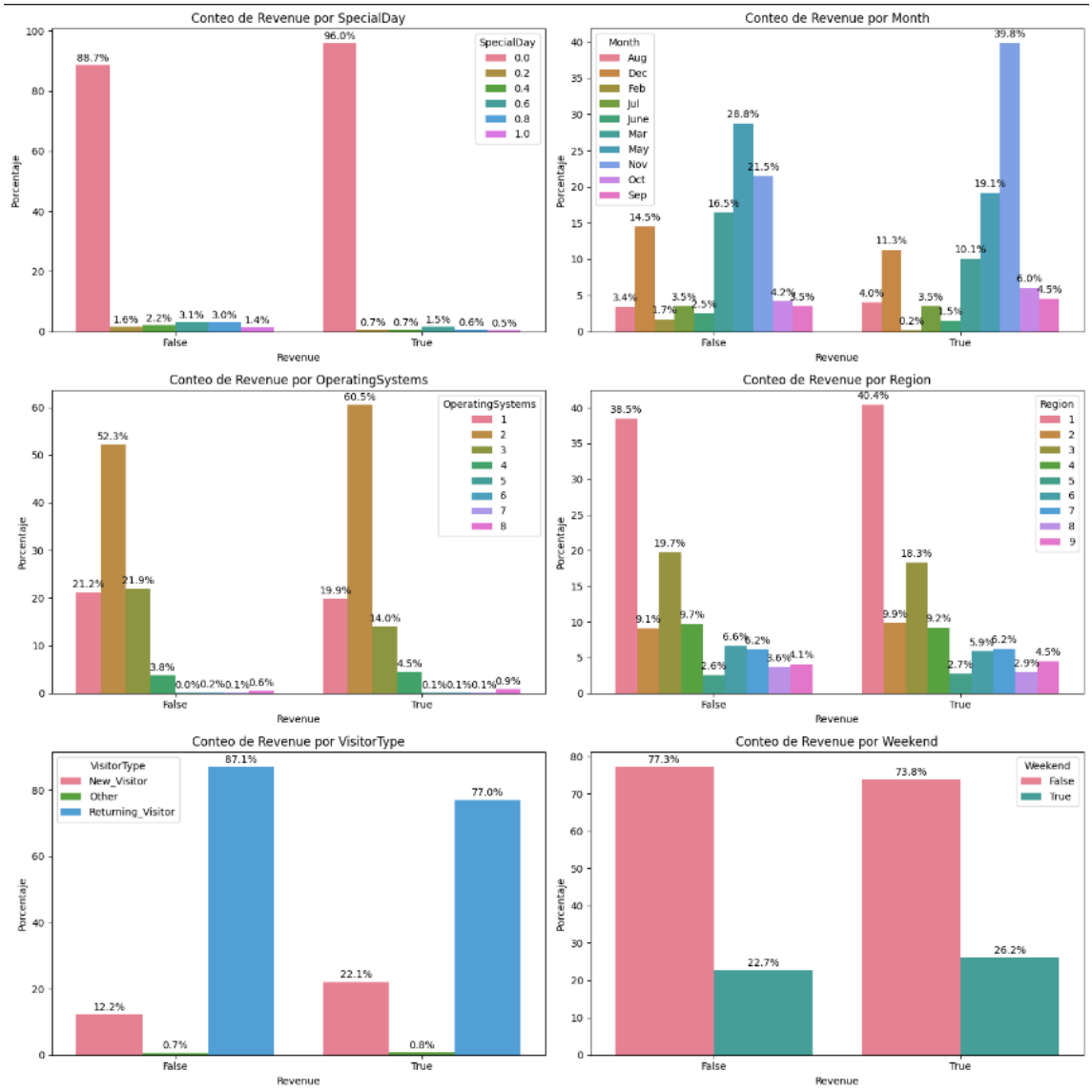
3 Análisis de los datos

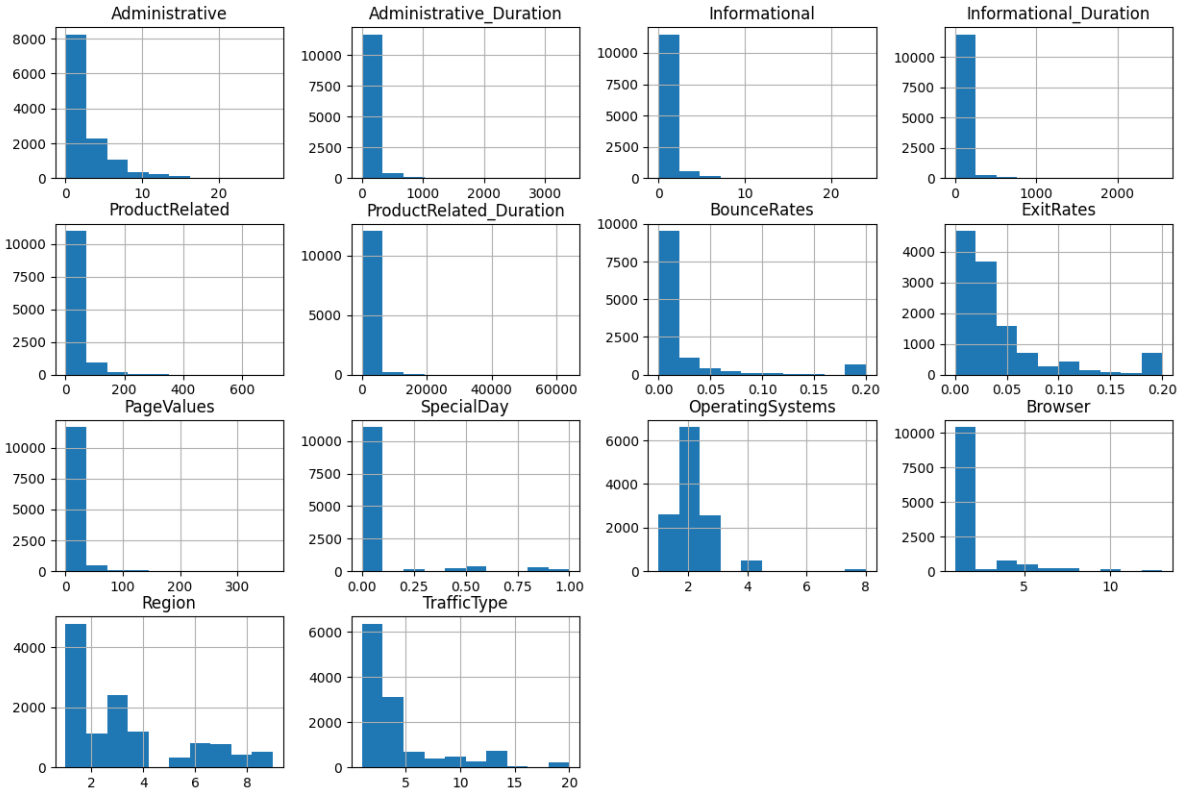
3.1 Descripción del Dataset

El dataset utilizado en este estudio proviene de la **UC Irvine Machine Learning Repository**, específicamente diseñado para el análisis de la intención de compra en una tienda online. Incluye información de **12,330 sesiones** registradas en un sitio de comercio electrónico durante un período de un año. Los datos están estructurados para representar comportamientos individuales de los usuarios y permitir la predicción de si una sesión concluirá en una compra.

3.2 Características del Dataset

El dataset contiene un total de **17 atributos**, que se dividen en variables numéricas y categóricas, además de la variable objetivo.





1. **Variables Numéricas (10)**: Estas variables proporcionan información cuantitativa sobre el comportamiento de los usuarios en la plataforma. Entre las más relevantes se encuentran:

   - **Administrative**: Número de páginas relacionadas con tareas administrativas visitadas durante la sesión.
   - **Informational**: Número de páginas relacionadas con información general visitadas.
   - **ProductRelated**: Número de páginas de productos vistas durante la sesión.
   - **Bounce Rate**: Tasa de rebote, indica la proporción de usuarios que abandonaron el sitio después de ver una sola página.
   - **Exit Rate**: Tasa de salida, mide la proporción de usuarios que abandonaron el sitio desde una página específica.
   - **Page Value**: Valor promedio asignado a una página en función de las conversiones generadas.

2. **Variables Categóricas (7)**: Estas variables describen características cualitativas de las sesiones, como el tipo de visitante o el período en que se llevó a cabo la actividad.

   - **Month:** Mes del año en que tuvo lugar la sesión (e.g., Jan, Feb).
   - **OperatingSystems:** Sistema operativo del usuario.
   - **Browser:** Navegador utilizado durante la sesión.
   - **Region:** Región geográfica del usuario.
   - **TrafficType:** Tipo de tráfico que generó la visita (e.g., tráfico directo, referidos).
   - **VisitorType:** Indica si el usuario es recurrente, nuevo o pertenece a otra categoría.

- **Weekend:** Variable booleana que indica si la sesión ocurrió en un fin de semana.

3. **Variable Objetivo (1)**:
   - **Revenue**: Variable binaria que indica si la sesión resultó en una compra (1) o no (0).

Columna	Descripción
Revenue	Indica si la sesión concluyó con una transacción; se puede usar como etiqueta de clase (target).
Administrative	Número de páginas de tipo administrativo visitadas por el usuario en esa sesión.
Administrative Duration	Tiempo total que el usuario pasó en páginas de tipo administrativo durante esa sesión.
Informational	Número de páginas de tipo informativo visitadas por el usuario en esa sesión.
Informational Duration	Tiempo total que el usuario pasó en páginas de tipo informativo durante esa sesión.
Product Related	Número de páginas relacionadas con productos visitadas por el usuario en esa sesión.
Product Related Duration	Tiempo total que el usuario pasó en páginas relacionadas con productos durante esa sesión.
Bounce Rate	Porcentaje de visitantes que ingresaron al sitio desde una página específica y luego abandonaron el sitio sin realizar más interacciones.
Exit Rate	Porcentaje de visitas en las que una página específica fue la última en la sesión, calculado sobre el total de vistas de esa página.
Page Value	Valor promedio de una página web que un usuario visitó antes de completar una transacción en el sitio de comercio electrónico.
Special Day	Cercanía de la visita a un día especial (ej., Día de la Madre, San Valentín), en el cual es más probable que la sesión termine en una transacción. Se basa en la dinámica del comercio electrónico, como la proximidad entre la fecha de orden y la de entrega. Toma un valor máximo de 1 en el día con más relevancia para el evento especial (ej., 8 de febrero para San Valentín).
Operating System	Sistema operativo del dispositivo utilizado durante la sesión.
Browser	Navegador utilizado durante la sesión.
Region	Región geográfica del usuario durante la sesión.
Traffic Type	Tipo de tráfico que generó la visita (por ejemplo, orgánico, de referencia, directo).
Visitor Type	Tipo de visitante: si es un visitante recurrente o uno nuevo.
Weekend	Valor booleano que indica si la visita ocurrió durante el fin de semana.
Month	Mes del año en el que ocurrió la sesión.

3.3 Aspectos Importantes del Dataset

1. **Desbalance de Clases**:
   - El 84.5% de las sesiones no culminaron en compra (Revenue = 0).
   - Solo el 15.5% de las sesiones resultaron en una compra (Revenue = 1).

```
Revenue
False    84.525547
True     15.474453
Name: proportion, dtype: float64
Revenue
False    10422
True      1908
Name: count, dtype: int64
```

2. **Calidad de los Datos**:
   - El dataset está bien estructurado y no presenta valores faltantes.
   - Las características categóricas requieren transformación, como codificación (e.g., One-Hot Encoding), para ser utilizadas en modelos de Machine Learning.

3. **Propósito del Dataset**: Este dataset está diseñado para:
   - Clasificación binaria: predecir si una sesión generará ingresos.
   - Análisis exploratorio: identificar factores clave que afectan la intención de compra.

- Optimización de campañas: comprender qué características de las sesiones pueden guiar estrategias de marketing personalizadas.

	COL_N	DATA_TYPE	NO MISSING	MISSING	MISSING (%)	UNIQUE_VALUES	CARDIN (%)	DATA_CLASS
0	Administrative	int64	12330	0	0.0	27	0.22	Numérica Discreta
1	Administrative_Duration	float64	12330	0	0.0	3335	27.05	Numérica Discreta
2	Informational	int64	12330	0	0.0	17	0.14	Numérica Discreta
3	Informational_Duration	float64	12330	0	0.0	1258	10.20	Numérica Discreta
4	ProductRelated	int64	12330	0	0.0	311	2.52	Numérica Discreta
5	ProductRelated_Duration	float64	12330	0	0.0	9551	77.46	Numérica Continua
6	BounceRates	float64	12330	0	0.0	1872	15.18	Numérica Discreta
7	ExitRates	float64	12330	0	0.0	4777	38.74	Numérica Continua
8	PageValues	float64	12330	0	0.0	2704	21.93	Numérica Discreta
9	SpecialDay	float64	12330	0	0.0	6	0.05	Categórica
10	Month	object	12330	0	0.0	10	0.08	Numérica Discreta
11	OperatingSystems	int64	12330	0	0.0	8	0.06	Categórica
12	Browser	int64	12330	0	0.0	13	0.11	Numérica Discreta
13	Region	int64	12330	0	0.0	9	0.07	Categórica
14	TrafficType	int64	12330	0	0.0	20	0.16	Numérica Discreta
15	VisitorType	object	12330	0	0.0	3	0.02	Categórica
16	Weekend	bool	12330	0	0.0	2	0.02	Binaria
17	Revenue	bool	12330	0	0.0	2	0.02	Binaria

4 Reducción de Características

En el proceso de reducción de dimensionalidad del dataset, se aplicaron diversas técnicas para identificar y seleccionar las características más relevantes, eliminando aquellas que no aportan información significativa al modelo o que introducen ruido. A continuación, se describe el procedimiento y los resultados obtenidos:

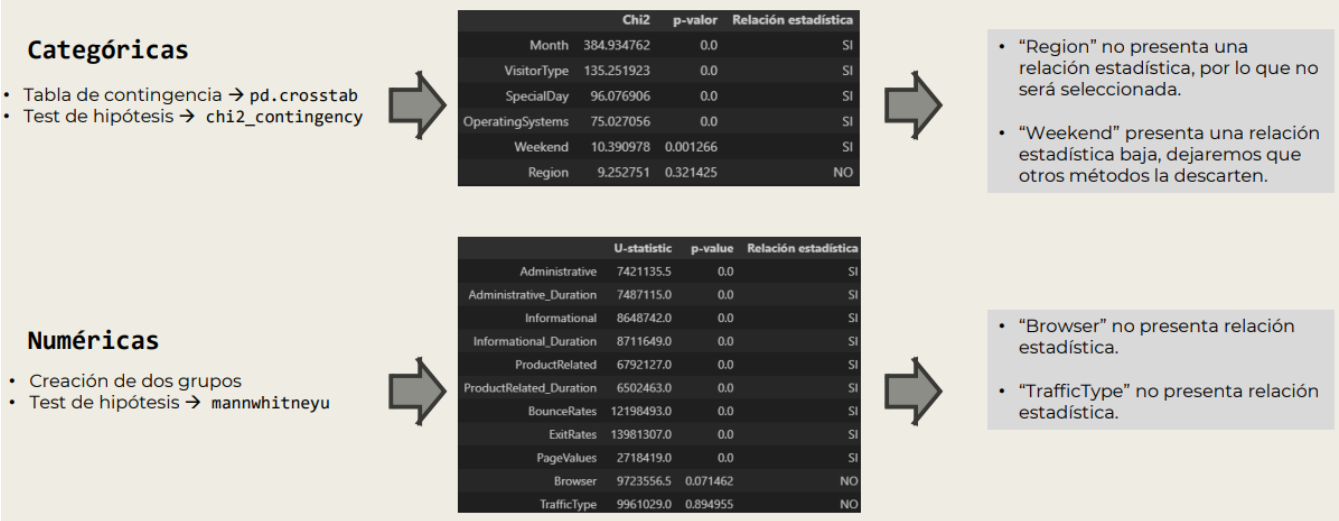
4.1 Justificación de la Reducción de Características

- **Evitar el ruido:** Muchas variables pueden no estar relacionadas con la variable objetivo (Revenue), lo que podría afectar negativamente al rendimiento del modelo.
- **Reducir el costo computacional:** Menos variables implican modelos más rápidos y eficientes.
- **Mejorar la interpretabilidad:** Una menor cantidad de características facilita la comprensión de los factores más relevantes.

4.2 Métodos Utilizados

Se implementaron varias técnicas para evaluar la importancia de las características, tanto desde un enfoque estadístico como utilizando modelos predictivos. Estas fueron:

1. **Pruebas de Hipótesis**:
   - **Numéricas:** Se utilizó el test U de Mann-Whitney para evaluar si las distribuciones de las características numéricas difieren significativamente entre las clases (Revenue).
   - **Categóricas:** Se aplicó Chi-cuadrado para medir la asociación entre las variables categóricas y la variable objetivo.



Resultados:

- Variables como Region y Browser no mostraron una relación estadísticamente significativa con el target, por lo que fueron descartadas.
- Variables como Bounce Rate y VisitorType mostraron una fuerte asociación.

2. **Feature Importance con XGBoost**:
   - Se entrenaron modelos XGBoost utilizando:
     - Balanceo de clases con scale_pos_weight.
     - Under-sampling de la clase mayoritaria.
     - SMOTE para sobremuestreo de la clase minoritaria.
   - La importancia de las características se calculó a partir de las ganancias proporcionadas por el modelo.

Resultados:

- Page Value, Bounce Rate, y Exit Rate fueron las variables numéricas más importantes.
- Las transformaciones como SMOTE permitieron detectar un mayor número de compradores potenciales.

3. **ANOVA y Mutual Information**:

   - Se seleccionaron características mediante SelectKBest utilizando pruebas de ANOVA y la métrica de información mutua.

   - Ayudaron a identificar las características con mayor capacidad predictiva.

4. **Selección Secuencial y Eliminación Recursiva**:

   - **SFS (Sequential Feature Selection):** Seleccionó las mejores 10 características mediante un proceso iterativo.

   - **RFE (Recursive Feature Elimination):** Evaluó la importancia relativa de las características y eliminó las menos relevantes.

5. **Hard Voting**:

   - Se combinó la salida de las técnicas anteriores con un enfoque de votación para identificar las características seleccionadas con mayor consistencia.

   Resultados:

   - Se observaron saltos significativos entre las características más repetidas en las votaciones, definiendo un conjunto final de características.

4.3 Conjunto Final de Características Seleccionadas

Después de aplicar todas las técnicas, las características finales seleccionadas fueron:

1. SpecialDay
2. Month
3. PageValues
4. ProductRelated
5. VisitorType
6. Administrative
7. ProductRelated_Duration
8. BounceRates
9. ExitRates

Estas características proporcionaron el mejor balance entre la capacidad predictiva y la simplicidad del modelo.

4.4 Impacto de la Reducción de Características y SMOTE

En este apartado se evalúa cómo las técnicas de reducción de características y balanceo de clases impactan en el rendimiento del modelo predictivo. Para ello, se compararon los resultados de un modelo baseline con los obtenidos al aplicar **reducción de características**, **SMOTE** y la combinación de ambas.

1. Modelo Baseline

El modelo baseline utiliza **XGBoost** con todos los atributos del dataset, sin aplicar técnicas de reducción ni balanceo de clases. Este sirve como punto de referencia para medir el impacto de las optimizaciones.

Resultados del Baseline:

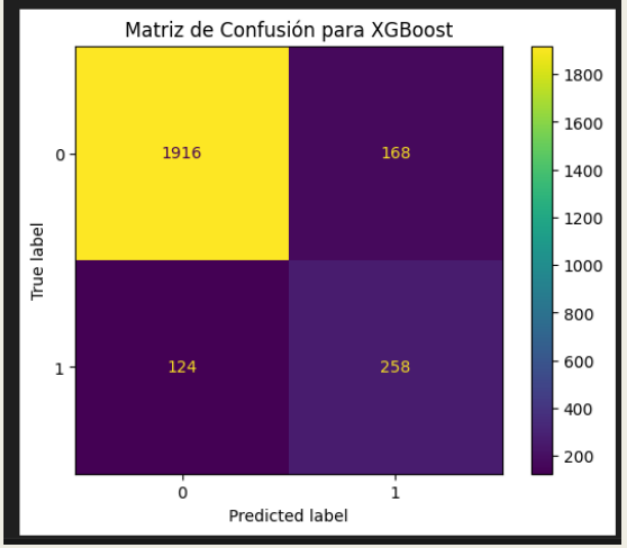
- **F1-Score**: Estándar.
- **Recall de la Clase 1**: 0.68.

Conclusión: El modelo detecta compradores potenciales, pero su capacidad está limitada por el ruido en las características y el desbalanceo de clases.

```python
modelo_xgb = xgb.XGBClassifier(enable_categorical=True, use_label_encoder=False, eval_metric='aucpr',
                               scale_pos_weight=peso_clases, random_state=42)

modelo_xgb.fit(X_train, y_train)
```

```python
y_pred = modelo_xgb.predict(X_test)

# Obtener el classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))

# Mostrar la matriz de confusión
print("Confusion Matrix:")
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.title("Matriz de Confusión para XGBoost")
plt.show()
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.92      0.93      2084
           1       0.61      0.68      0.64       382

    accuracy                           0.88      2466
   macro avg       0.77      0.80      0.78      2466
weighted avg       0.89      0.88      0.88      2466

Confusion Matrix:
```



2. Baseline con Reducción de Características

Se entrenó el modelo utilizando únicamente las características seleccionadas tras aplicar técnicas de reducción como pruebas de hipótesis, importancia de características, y votación mayoritaria.

Resultados del Baseline con Feature Reduction:

- **F1-Score**: Similar al baseline, indicando que la reducción no compromete el desempeño general.
- **Recall de la Clase 1**: Mejora a 0.71.

Conclusión: La reducción de características elimina el ruido y simplifica el modelo, mejorando su capacidad para identificar compradores potenciales sin aumentar el riesgo de overfitting.

3. Baseline con SMOTE

Para abordar el desbalance de clases, se utilizó el algoritmo **SMOTE (Synthetic Minority Oversampling Technique)**, que genera nuevas instancias sintéticas de la clase minoritaria.

Resultados del Baseline con SMOTE:

- **F1-Score**: Mejorado debido al incremento de la sensibilidad hacia la clase minoritaria.
- **Recall de la Clase 1**: Incremento significativo a 0.77.

Conclusión: SMOTE es más efectivo que la reducción de características para mejorar el recall, ya que amplía el conjunto de compradores potenciales detectados, reduciendo el sesgo hacia la clase mayoritaria.

4. Baseline con Feature Reduction + SMOTE

Se combinaron ambas técnicas para aprovechar las ventajas de cada enfoque: reducción del ruido mediante selección de características y aumento de la sensibilidad mediante balanceo de clases.

Resultados del Baseline con Feature Reduction + SMOTE:

- **F1-Score**: Consistente con los modelos anteriores.
- **Recall de la Clase 1**: Incremento máximo a 0.79.

Conclusión: La combinación de técnicas obtiene el mejor desempeño, maximizando la capacidad del modelo para identificar compradores potenciales mientras mantiene un equilibrio en las métricas globales.

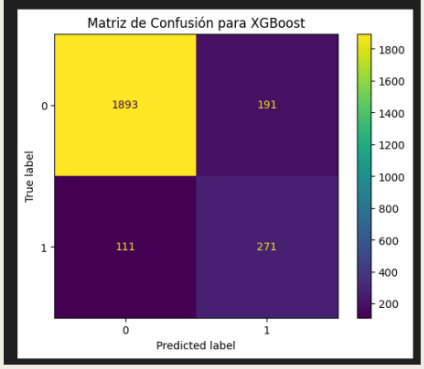
Modelo	F1-Score	Recall Clase 1
Baseline	Estándar	0.68
Baseline con Feature Reduction	Estándar	0.71
Baseline con SMOTE	Mejorado	0.77
Feature Reduction + SMOTE	Mejorado	0.79

Baseline con Feature Reduction

Classification Report:

	precision	recall	f1-score	support
0	0.94	0.91	0.93	2084
1	0.59	0.71	0.64	382
accuracy			0.88	2466
macro avg	0.77	0.81	0.78	2466
weighted avg	0.89	0.88	0.88	2466

Confusion Matrix:

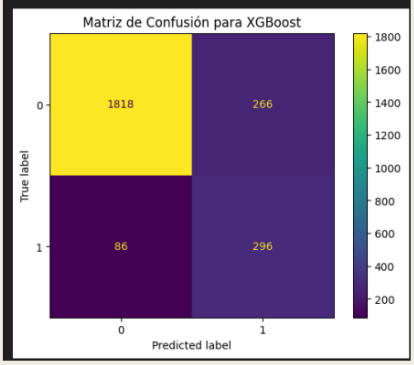


Baseline con SMOTE

Classification Report:

	precision	recall	f1-score	support
0	0.95	0.87	0.91	2084
1	0.53	0.77	0.63	382
accuracy			0.86	2466
macro avg	0.74	0.82	0.77	2466
weighted avg	0.89	0.86	0.87	2466

Confusion Matrix:

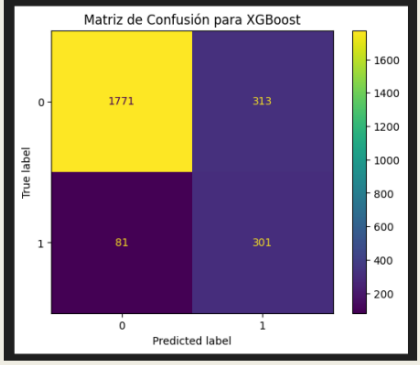


Baseline FR con SMOTE

Classification Report:

	precision	recall	f1-score	support
0	0.96	0.85	0.90	2084
1	0.49	0.79	0.60	382
accuracy			0.84	2466
macro avg	0.72	0.82	0.75	2466
weighted avg	0.88	0.84	0.85	2466

Confusion Matrix:



5 Machine Learning

El objetivo principal de esta sección es detallar la selección, implementación y evaluación de modelos de Machine Learning para la predicción de la intención de compra. En este proceso, se aplicaron métricas específicas para datasets desbalanceados, como **Balanced Accuracy** y **AUC-PR**, y un pipeline estructurado para garantizar la reproducibilidad del flujo de trabajo.

5.1 Pipeline

Se diseñó un pipeline que integra las siguientes etapas clave:

a) Transformación de Datos

Se implementaron **dos pipelines separados** para manejar las transformaciones específicas de las variables categóricas y numéricas:

1. **Pipeline para Variables Categóricas**:

- **OneHotEncoder**:
  - Este codificador transforma variables categóricas en un formato binario (1 o 0) para que puedan ser utilizadas por los modelos de Machine Learning.
  - Es ideal para manejar variables no ordinales como VisitorType o Month.

2. **Pipeline para Variables Numéricas**:

- **PowerTransformer**:
  - Se utilizó el método **Yeo-Johnson**, que ajusta la distribución de las variables numéricas para aproximarlas a una distribución normal.
  - Beneficios:
  - Reduce la asimetría (skewness) en los datos.
  - Mejora la capacidad de los modelos para manejar datos con grandes rangos o distribuciones no normales.
  - Aplicado en variables como Page Value y Bounce Rate.

b) Combinación de Pipelines

Para integrar las transformaciones anteriores, se utilizó un **ColumnTransformer**, que permite aplicar transformaciones específicas a subconjuntos de columnas del dataset. Este combina:

- El pipeline para variables categóricas.

- El pipeline para variables numéricas.
- Una transformación adicional para eliminar las características irrelevantes o redundantes mediante **Drop Features**.

Nota: Las características descartadas se identificaron previamente durante la reducción de características, como Region y Browser.

c) Pipeline para Manejo del Desbalanceo

Para abordar el desbalance de clases en el dataset, se añadió un pipeline específico utilizando **SMOTE** (Synthetic Minority Oversampling Technique) de la librería imbalanced-learn. Este pipeline genera instancias sintéticas de la clase minoritaria (Revenue = 1) durante el entrenamiento.

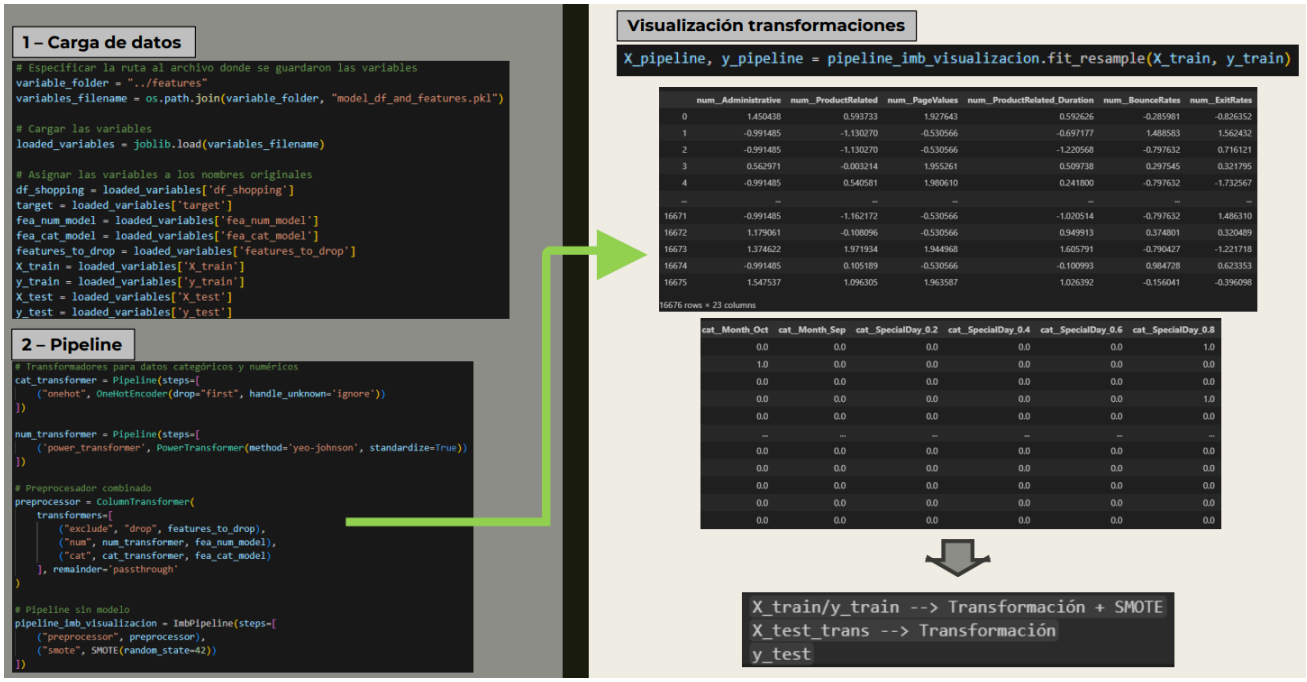
Ventajas de SMOTE:

- Amplía el número de ejemplos de la clase minoritaria, reduciendo el sesgo del modelo hacia la clase mayoritaria.
- Aumenta el recall de la clase minoritaria sin necesidad de descartar instancias de la clase mayoritaria.

d) Pipeline Completo

El pipeline final se estructura en los siguientes pasos:

1. **Transformaciones iniciales**:
   - Aplicación del **ColumnTransformer** que integra:
     - OneHotEncoder para variables categóricas.
     - PowerTransformer con método Yeo-Johnson para variables numéricas.
     - Drop Features para descartar variables irrelevantes.
2. **Manejo del desbalance**:
     - Aplicación de **SMOTE** mediante un pipeline de imbalanced-learn.
3. **Modelo de Machine Learning**:
     - El modelo se entrena sobre los datos transformados y balanceados, utilizando los algoritmos seleccionados (como SVM, Random Forest, XGBoost, etc.).



5.2 Modelos Implementados

El pipeline incluyó una amplia variedad de algoritmos de clasificación. A continuación, se listan los modelos probados con sus configuraciones iniciales:

- Random Forest
- XGBoost
- LightGBM
- CatBoost
- Logistic Regression
- SVM (Support Vector Machines)
- KNN (K-Nearest Neighbors)
- Naive Bayes
- Decision Tree
- Gradient Boosting
- AdaBoost

5.3 Validación Inicial con Cross-Validation

Se utilizó **Balanced Accuracy** como métrica para evaluar los modelos en la validación cruzada. Los resultados clasificaron los modelos de la siguiente forma:

Modelo	Balanced Accuracy
SVM	0.860600
Gradient Boosting	0.851546
Logistic Regression	0.845554
AdaBoost	0.839757
Random Forest	0.828036

3 – Modelos

```
models = {
    'RandomForest': RandomForestClassifier(class_weight='balanced', random_state=42, n_jobs=-1),
    'XGBoost': XGBClassifier(eval_metric='aucpr', random_state=42, n_jobs=-1),
    'LightGBM': LGBMClassifier(random_state=42, verbose=-1, n_jobs=-1),
    'CatBoost': CatBoostClassifier(verbose=0, random_state=42, thread_count=-1),
    'LogisticRegression': LogisticRegression(class_weight='balanced', random_state=42, n_jobs=-1),
    'SVM': SVC(class_weight='balanced', probability=True, random_state=42),
    'KNN': KNeighborsClassifier(n_neighbors=5, n_jobs=-1),
    'NaiveBayes': GaussianNB(),
    'DecisionTree': DecisionTreeClassifier(class_weight='balanced', random_state=42),
    'GradientBoosting': GradientBoostingClassifier(random_state=42, n_iter_no_change=10),
    'AdaBoost': AdaBoostClassifier(random_state=42)
}
```

4 – Pipeline + Cross validation + Best model

```
balanced_accuracy_scores = {}

for model_name, model in models.items():
    pipeline_imb = ImbPipeline(steps=[
    ("preprocessor", preprocessor),
    ("smote", SMOTE(random_state=42)),
    ('classifier', model)
    ])

    pipeline_imb.fit(X_train, y_train)

    models_pkl_folder = "../model/Base_model/Scoring_balanced_accuracy"
    model_filename = os.path.join(models_pkl_folder, f"{model_name}_base_model.pkl")
    joblib.dump(pipeline_imb, model_filename)
    print(f"Modelo {model_name} guardado como {model_filename}\n")

    cv_scores = cross_val_score(pipeline_imb, X_train, y_train, cv=5, scoring="balanced_accuracy", error_score="raise")
    mean_score = np.mean(cv_scores)
    balanced_accuracy_scores[model_name] = mean_score
    print(f"Modelo: {model_name}, Balanced Accuracy: {mean_score:.4f}")
```

	Modelo	Balanced Accuracy
0	SVM	0.860600
1	GradientBoosting	0.851546
2	LogisticRegression	0.845554
3	AdaBoost	0.839757
4	RandomForest	0.828036
5	LightGBM	0.825159
6	CatBoost	0.823112
7	KNN	0.822724
8	XGBoost	0.814518
9	DecisionTree	0.765163
10	NaiveBayes	0.649397

5.4 Mejora de hiperparámetros de los mejores modelos de Cross Validation

Se evaluaron hiperparámetros clave para cada modelo, como el número de estimadores (n_estimators), tasa de aprendizaje (learning_rate), regularización (C en Logistic Regression y SVM)., entre muchos otros.

5 – Hiperparámetros

```
param_grids = {
    'XGBoost': {
    'classifier__n_estimators': [50, 100, 200],
    'classifier__learning_rate': [0.01, 0.1, 0.2],
    'classifier__max_depth': [3, 5, 7],
    'classifier__scale_pos_weight': [1, 5, 10]
    },
    'SVM': {
        'classifier__C': [0.1, 1, 10],
        'classifier__kernel': ['linear', 'rbf'],
        'classifier__gamma': ['scale', 'auto']
    },
    'GradientBoosting': {
        'classifier__n_estimators': [50, 100, 200],
        'classifier__learning_rate': [0.01, 0.1, 0.2],
        'classifier__max_depth': [3, 5, 7]
    },
    'LightGBM': {
        'classifier__n_estimators': [50, 100, 200],
        'classifier__learning_rate': [0.01, 0.1, 0.2],
        'classifier__max_depth': [-1, 3, 5, 7],
        'classifier__num_leaves': [20, 31, 50]
    },
    'LogisticRegression': {
        'classifier__C': [0.01, 0.1, 1, 10, 100],
        'classifier__penalty': ['l2', 'none'],
        'classifier__solver': ['lbfgs', 'saga']
    },
    'AdaBoost': {
        'classifier__n_estimators': [50, 100, 200],
        'classifier__learning_rate': [0.01, 0.1, 0.5, 1.0]
    }
}
```

6 – Grid Search

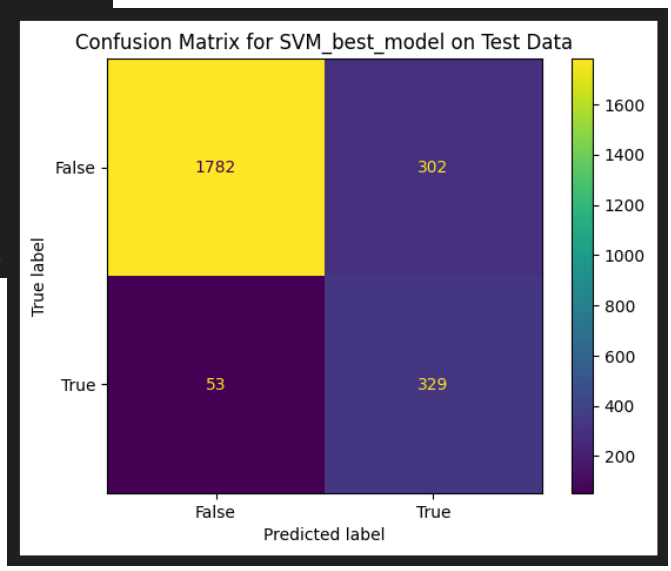
```
grid_search_results = []

# GridSearch para cada modelo
for model_name, model in models.items():
    pipeline_imb = ImbPipeline(steps=[
    ("preprocessor", preprocessor),
    ("smote", SMOTE(random_state=42)),
    ('classifier', model)
    ])


    grid_search = GridSearchCV(
        pipeline_imb,
        param_grid=param_grids[model_name],
        cv=5,
        scoring="balanced_accuracy",
        n_jobs=-1
    )

    grid_search.fit(X_train, y_train)

    best_model = grid_search.best_estimator_
    best_score = grid_search.best_score_
    grid_search_results.append((model_name, best_model, best_score))
    print(f"Modelo: {model_name}, Mejor Balanced Accuracy en validación: {best_score:.4f}")

    models_pkl_folder = "../model/Best_params_model/Scoring_balanced_accuracy"
    model_filename = os.path.join(models_pkl_folder, f"{model_name}_best_model.pkl")
    joblib.dump(best_model, model_filename)
    print(f"Modelo {model_name} guardado como {model_filename}\n")
```

5.5 Resultados del Modelo Ganador

Tras la optimización, **SVM** resultó ser el modelo con mejor desempeño. A continuación, se muestran las métricas clave obtenidas en el conjunto de prueba:

- **F1-Score**: se obtiene 0.65. Mejor que el 0.60 obtenido con baseline + FR + SMOTE
- **Recall de la Clase 1**: Incremento en 7 puntos sobre el mejor, obteniendo 0.86.

```
Resultados para el modelo: SVM_best_model
Balanced Accuracy en el conjunto de prueba: 0.8582

Reporte de Clasificación en el conjunto de prueba:
              precision    recall  f1-score   support

       False       0.97      0.86      0.91      2084
        True       0.52      0.86      0.65       382

    accuracy                           0.86      2466
   macro avg       0.75      0.86      0.78      2466
weighted avg       0.90      0.86      0.87      2466

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



5.6 Conclusiones del Modelo de Machine Learning

1. **Pipeline Eficiente:** El uso de un pipeline estructurado con Imbalanced-learn y Cross-Validation aseguró la robustez del flujo de trabajo.

2. **SVM como Mejor Modelo:** La combinación de hiperparámetros optimizados y el enfoque en Balanced Accuracy permitió a SVM destacar sobre otros algoritmos.

3. **Uso de Métricas Apropiadas:** La utilización de **Balanced Accuracy** y **AUC-PR** fue crucial para abordar el desbalance de clases y garantizar una evaluación precisa.

4. **Resultados:**

    - El recall del 86% para la clase de compradores potenciales asegura que la mayoría de los usuarios con intención de compra sean detectados.

Este modelo puede integrarse en tiempo real para mejorar la estrategia de marketing personalizada en la tienda online de Manuel, maximizando la tasa de conversión y optimizando los recursos.

6 Deep Learning

En este trabajo, aunque los proyectos de predicción en comercio electrónico suelen basarse en modelos de Machine Learning, también se implementó una red neuronal para explorar su desempeño en la predicción de la intención de compra. Esto permitió analizar el impacto de una arquitectura más compleja en comparación con los modelos tradicionales de Machine Learning.

6.1 Justificación de la Red Neuronal

1. **Uso de Machine Learning en Comercio Electrónico**:
    - Los modelos de Machine Learning como Random Forest, SVM y XGBoost suelen ser eficaces y eficientes en este tipo de problemas, pero su desempeño puede verse limitado en problemas más complejos o no lineales.
2. **Motivación para el Uso de Deep Learning**:
    - La red neuronal puede capturar relaciones más complejas entre las variables, gracias a su capacidad para modelar funciones no lineales.
    - Aunque no es habitual para este tipo de proyectos, se quiso explorar si una arquitectura más compleja podía aportar ventajas adicionales.

6.2 Fases del Desarrollo del Modelo

1. **Carga y Transformación de los Datos**:
    - **Preprocesamiento**:
        - Se estandarizaron los datos para que estuvieran en un rango adecuado para la red neuronal. Se utilizó un escalado entre **-1 y 1**.
        - Esto asegura que las entradas tengan magnitudes similares, mejorando la estabilidad del entrenamiento.
        - Los datos se dividieron en conjuntos de entrenamiento y prueba, manteniendo la proporción de clases original.

```
scaler = MinMaxScaler(feature_range=(-1, 1))
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

2. **Arquitectura de la Red Neuronal**:
    - **Estructura**:
        - **Capa de Entrada**:
            - Número de neuronas igual al número de características seleccionadas tras la reducción.
        - **Capas Ocultas**:
            - Dos capas ocultas con 128 y 64 neuronas, respectivamente.
            - Activación **ReLU** en ambas capas para capturar relaciones no lineales.
        - **Capa de Salida**:
            - Una única neurona con activación **sigmoide** para predecir la probabilidad de la clase Revenue = 1.
    - **Optimización**:
        - Función de pérdida: **Binary Crossentropy**, adecuada para problemas de clasificación binaria.
        - Optimizador: **Adam**, debido a su eficiencia en ajustes dinámicos de la tasa de aprendizaje.

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	2,944
batch_normalization (BatchNormalization)	(None, 128)	512
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
batch_normalization_1 (BatchNormalization)	(None, 64)	256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 32)	2,080
dense_3 (Dense)	(None, 1)	33

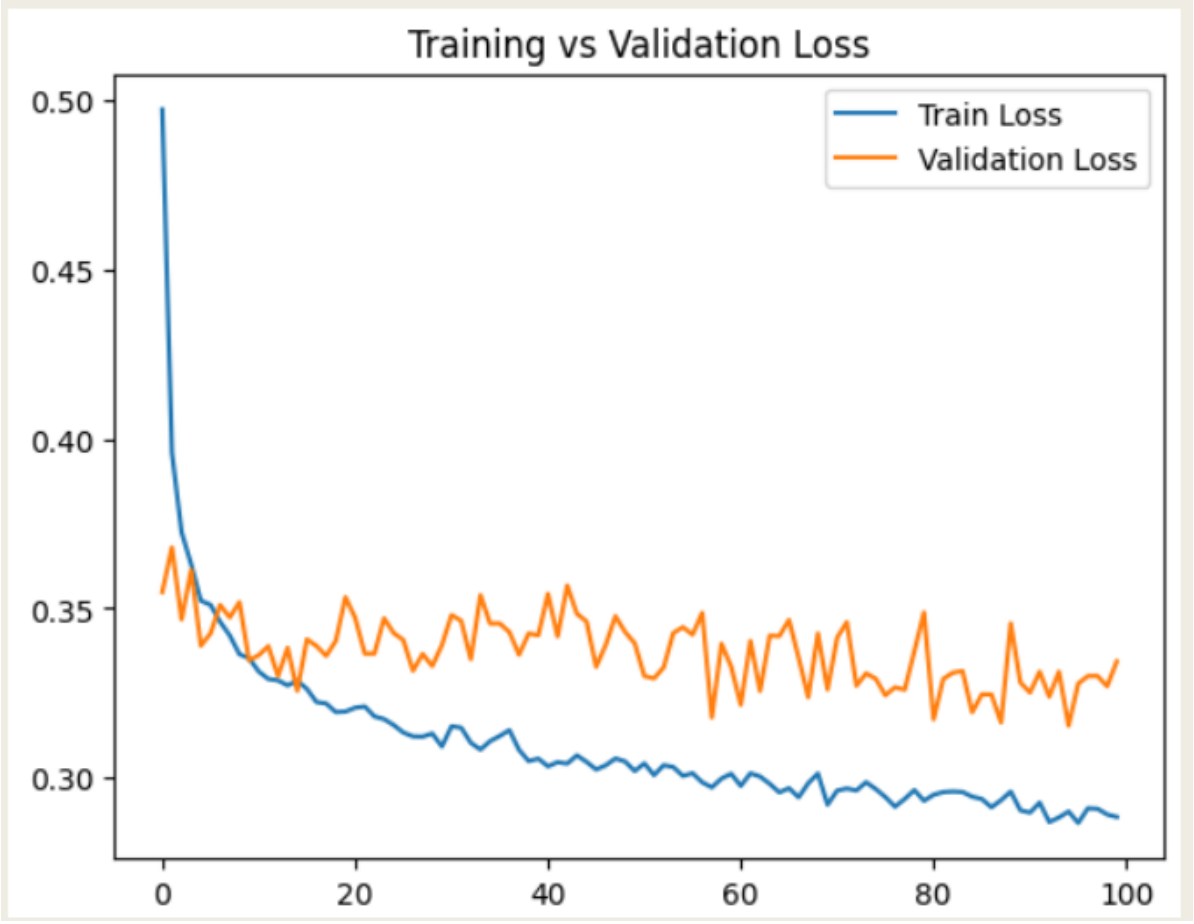
3. **Entrenamiento del Modelo**:

   - **Primer Experimento: 100 Épocas Forzadas**:

     - Se entrenó el modelo durante 100 épocas sin aplicar early stopping.
     - Observaciones:
     - La función de pérdida en el conjunto de validación se estancó y, en algunas ocasiones, incluso aumentó.
     - Esto es indicativo de **overfitting**, ya que el modelo comenzó a memorizar los datos de entrenamiento en lugar de generalizar.

     Resultado:

     - Aunque la matriz de confusión mostró buenos resultados en el conjunto de entrenamiento, esto no asegura un buen desempeño en datos nuevos.



- **Segundo Experimento: 100 Épocas con Patience = 10**:

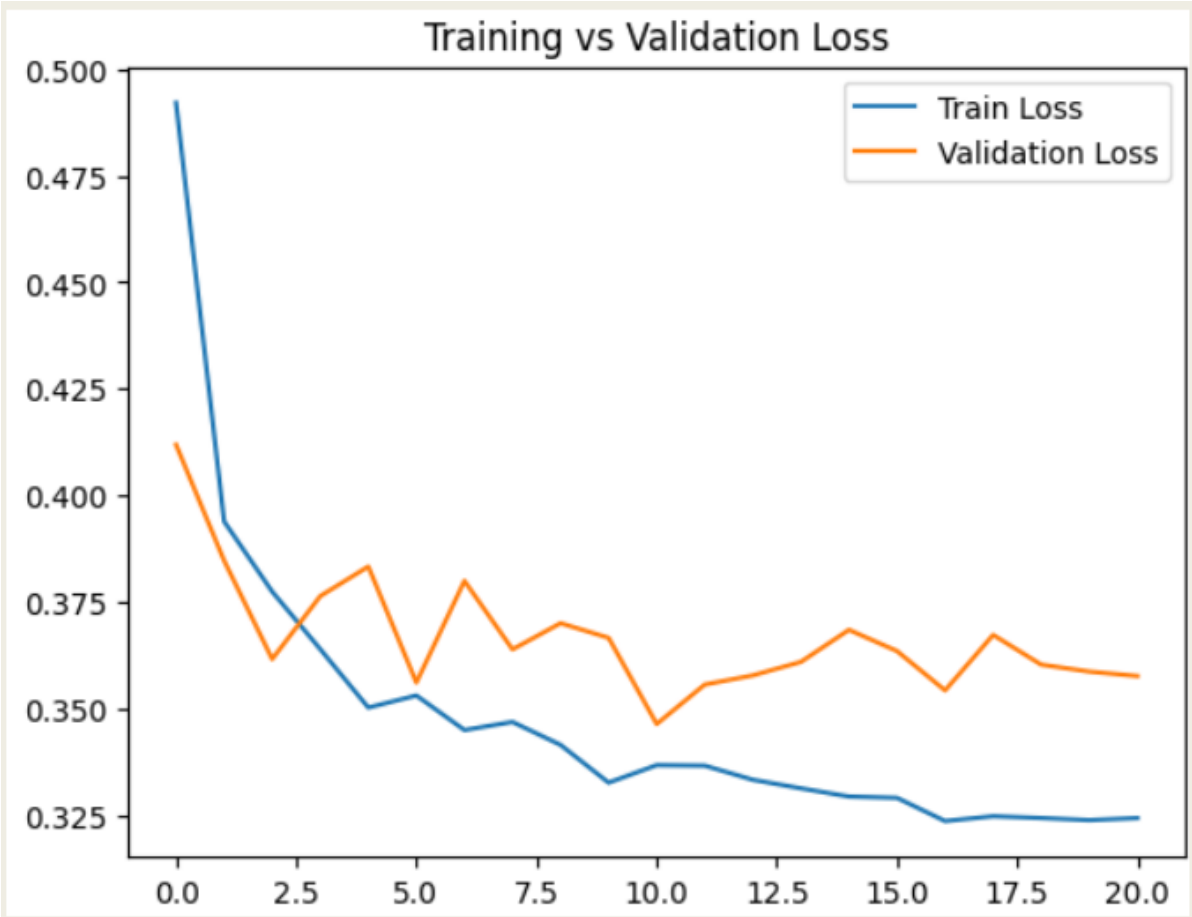
    - En este entrenamiento, se aplicó **early stopping** con un patience de 10 épocas, es decir, el modelo detenía el entrenamiento si no mejoraba la función de pérdida en 10 épocas consecutivas.

Observaciones:

- El entrenamiento se detuvo en la **época 11**, donde la función de pérdida alcanzó su mínimo.
- La función de pérdida en validación tuvo un comportamiento mucho más estable, evitando picos y caídas.

Resultado:

- La matriz de confusión mostró un desempeño ligeramente inferior en términos de predicción directa, pero el modelo probablemente **generalizó mejor**, siendo menos propenso a errores en datos nuevos.



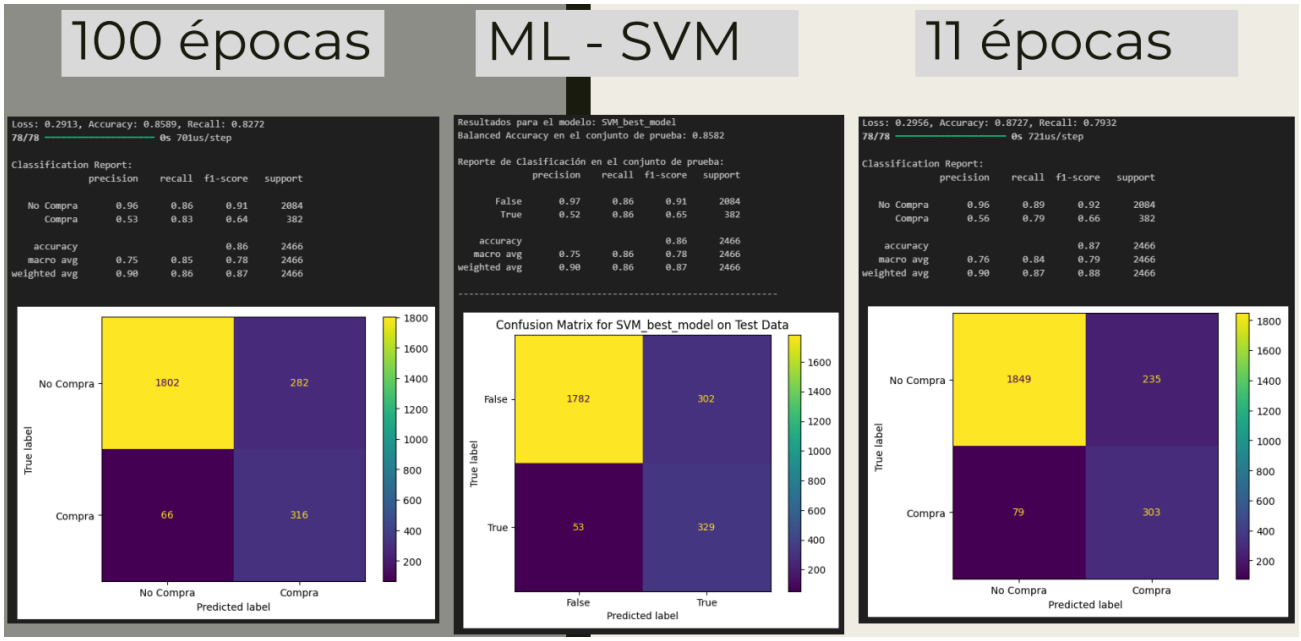
4. **Resultados y Comparativa**

   a) **Entrenamiento con 100 Épocas Forzadas**:

   - La función de pérdida en validación muestra un comportamiento errático, con picos y estancamientos.
   - La matriz de confusión indicó buenos resultados en el conjunto de prueba, pero hay alta probabilidad de que el modelo esté sobreajustado.

   b) **Entrenamiento con Early Stopping (Patience = 10)**:

   - La función de pérdida fue más estable, con una mejor relación entre entrenamiento y validación.
   - Aunque los resultados en la matriz de confusión fueron inferiores, se espera que este modelo tenga mejor capacidad de generalización.



5. **Conclusiones sobre el Uso de Deep Learning**

   a) **Ventajas del Modelo**:

   - El modelo de red neuronal permitió explorar relaciones más complejas entre las características.
   - Mostró resultados prometedores, aunque no superó significativamente a los modelos de Machine Learning en este caso.

   b) **Limitaciones**:

   - Los problemas de sobreajuste se hicieron evidentes en el primer experimento con 100 épocas.
   - Para datasets pequeños y con datos tabulares, como este, el Machine Learning sigue siendo más eficiente.

c) **Resultados Finales**:

- El modelo con early stopping es más confiable, pues generaliza mejor.
- Sin embargo, la simplicidad y eficiencia de los modelos de Machine Learning los hace más adecuados para este caso práctico.

La implementación del modelo de Deep Learning ha proporcionado insights valiosos, pero refuerza la idea de que para problemas tabulares como este, los modelos de Machine Learning son generalmente más efectivos y prácticos.

7 Probabilidades del modelo – Aplicaciones reales

En esta etapa, se entrenó un modelo SVM utilizando el argumento probability=True, permitiendo que el modelo genere no solo predicciones binarias (clase 0 o clase 1), sino también probabilidades asociadas a cada clase. Este enfoque abre un abanico de posibilidades para personalizar la estrategia de marketing en tiempo real, optimizando el impacto de las promociones.

7.1 Justificación del Uso de Probabilidades

1. **Limitaciones de las Predicciones Binarias**:
   - Con etiquetas binarias (0 o 1), no se tiene información sobre la confianza del modelo en cada predicción.
   - Esto limita la capacidad de priorizar estrategias según la cercanía de un usuario a realizar una compra.
2. **Ventajas de las Probabilidades**:
   - Las probabilidades ofrecen un nivel de granularidad que permite interpretar qué tan cerca lejos está un usuario de pertenecer a una clase específica.
   - Esto es esencial para personalizar las acciones de marketing, adaptándolas al comportamiento y la intención del cliente.

7.2 Entrenamiento del Modelo

1. **Configuración**:
   - El modelo SVM se entrenó con probability=True para habilitar la salida probabilística.
   - Esto requirió realizar un proceso adicional de ajuste de probabilidades mediante **Platt Scaling**, que SVM implementa internamente.

```
svm_model = SVC(probability=True, class_weight='balanced', random_state=42)
```

2. **Resultados**:

   - Las probabilidades generadas por el modelo proporcionan una confianza en el rango [0, 1] para cada clase (Revenue = 0 y Revenue = 1).

```
# Hacer predicciones con probabilidades del mejor modelo
y_pred_proba = best_model.predict_proba(X_test)
print(f"Probabilidades predichas con el mejor modelo:\n{y_pred_proba}")
```

✓ 0.4s

```
Probabilidades predichas con el mejor modelo:
[[0.98981691 0.01018309]
 [0.88195589 0.11804411]
 [0.09650364 0.90349636]
 ...
 [0.12597906 0.87402094]
 [0.90906572 0.09093428]
 [0.83292412 0.16707588]]
```

7.3 Aplicaciones de las Probabilidades en Tiempo Real

Con las probabilidades, el modelo puede integrarse como un **input en un sistema automatizado de toma de decisiones**, permitiendo que las promociones se adapten dinámicamente al estado del cliente. A continuación, se describen ejemplos de cómo estas probabilidades pueden influir en la estrategia de marketing:

1. **Usuarios Casi Decididos (Probabilidad: 75%-90%)**:

   - Usuarios en este rango están próximos a realizar una compra, pero aún necesitan un empujón.

     Acción: Enviar promociones que aseguren la conversión, como:

     - Envío gratuito en la siguiente compra.
     - Descuento del 10% para compras superiores a cierta cantidad.

2. **Usuarios en Riesgo de Perderse (Probabilidad: 60%-75%)**:

   - Usuarios que muestran interés, pero fluctúan en su intención de compra (por ejemplo, suben al 70% y vuelven al 60%).

     Acción: Enviar promociones más agresivas para asegurar la conversión, como:

     - Descuento directo en el producto más visitado.
     - Cupón de 20% si finalizan la compra en menos de 24 horas.

3. **Usuarios Altamente Probables (Probabilidad: 85%-90% o Superior)**:

   - Usuarios con alta probabilidad de comprar, que ya están convencidos en su mayoría.

     Acción: Ofrecer promociones moderadas para aumentar el ticket promedio, como:

- Descuento del 5% al superar un monto específico.
- Promoción de envío gratuito si agregan un producto adicional al carrito.

7.4 Ventajas Futuras

1. **Estrategias en Tiempo Real**:
    - Las probabilidades generadas por el modelo pueden integrarse en un sistema de **toma de decisiones dinámico** que evalúe el estado del cliente en tiempo real o casi real.
    - Esto permitiría enviar promociones personalizadas basadas en la probabilidad actual, maximizando la tasa de conversión.
2. **Optimización de Recursos**:
    - Ajustar las promociones según la probabilidad evita gastos innecesarios en usuarios que probablemente ya comprarían sin incentivos adicionales.
    - Permite enfocar recursos en los usuarios con mayor riesgo de abandonar la compra.
3. **Adaptación Dinámica**:
    - Un sistema basado en probabilidades puede monitorizar cambios en el comportamiento del cliente (por ejemplo, si la probabilidad de compra disminuye), activando promociones específicas para revertir esta tendencia.

7.5 Conclusión

Entrenar el modelo SVM con probability=True abre la puerta a un sistema de marketing más sofisticado, donde las probabilidades actúan como una métrica clave para tomar decisiones. Esto no solo mejora la personalización de las promociones, sino que también optimiza el uso de los recursos de la empresa al enfocar esfuerzos en los clientes con mayor probabilidad de compra, maximizando tanto la conversión como los ingresos.