

A thick black L-shaped frame is positioned around the central text. It starts at the top left, goes right, then down, then right again, and finally down to the bottom right corner.

PREDICCIÓN DE COMPRA EN UN COMERCIO ON-LINE

INTRODUCCIÓN & DATOS CLAVE



Introducción



- Manuel ha montado un comercio on-line y desde hace un tiempo esta pensando en repartir promociones a los usuarios que entren en su web.
- El problema es que desconoce de antemano que usuarios son potenciales compradores y cuales no.
- Tiene pensado campañas de marketing diferentes, pero claro, estas tienen costes diferentes también, por lo que necesita tener claro a que tipo de cliente se las envía.
- La idea que Manuel nos ha transmitido es que le gustaría que en tiempo real se supiera lo cerca o lejos que están esas personas de comprar, para así usar marketing aún más personalizado.

¡Que buenas ideas!...pero...
¿como se ha quien enviárselas?



Marketing

Potenciales Compradores

- Envío gratuito en la siguiente compra.
- Descuento del 10% en la próxima compra.
- Devoluciones gratis por compras superiores a 100€.

Compradores No Potenciales

- Primera compra: 1 producto gratis (seleccionados).
- Si lo encuentras más barato, te damos un cupón con la diferencia.
- Compra y paga después.

Fidelización

- Acceso a programa de puntos por cada compra.
- Beneficios exclusivos por ser cliente frecuente (ej. descuentos recurrentes).

- Registro gratuito en el programa de fidelización con beneficios iniciales.
- Promociones por primera compra.

Promociones

- Ofertas personalizadas basadas en el historial de navegación o compras.
- Venta cruzada con productos relacionados.

- Cupones de descuento directo para incentivar la primera compra.
- Acceso a un descuento exclusivo por tiempo limitado.

Comunicación

- Email personalizado con recomendaciones específicas según su comportamiento.
- Notificaciones de productos populares en oferta.

- Campañas de email informativas sobre ventajas del servicio.
- Redes sociales: contenidos que generen confianza en la marca.

Experiencia de Usuario

- Envío premium gratis durante 6 meses.
- Acceso prioritario a ofertas y productos exclusivos.

- Asistencia personalizada para resolver dudas antes de la compra.
- Tutoriales o guías que destaquen ventajas del producto.

Retargeting

- Recordatorio de carrito abandonado con ofertas específicas.

- Retargeting mediante redes sociales y anuncios dinámicos destacando ventajas clave o promociones.

“Manuel se acordó que su amigo Luis estaba estudiando para ser Data Science...y pensó... ¿Porque no comentárselo a ver que puede hacer con los datos?”

“Luis le ha comentado muchas veces las posibilidades que tiene el Machine Learning...así que...vamos a decírselo!!!”



Datos clave

- Manuel ha recopilado en estos años datos de más de 12.000 clientes.
- Estos clientes han sido etiquetados como Compradores y No compradores.
- Se han obtenido 17 características del patrón de conducta de estos usuarios.

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.



Objetivo

Tras hablar con Manuel, su idea es:

- No dejar escapar a los potenciales compradores.
- Esto viene a decir que quiere que estos reciban si o si, los descuentos de su clase.
- Nos comenta que no le importa que le demos descuentos de la Clase “comprador” a la Clase “comprador no potencial”, pues tampoco perjudica, aunque estos descuentos están más de la mano de la fidelización que para realizar una primera compra.

Con todo ello, se estable como métrica de control el **recall de la clase 1**.

Aunque, nos vamos a centrar en mejorar el **balance accuracy** en vez del recall de la clase 1 directamente, pues es muy probable que haya un **overfitting más acusado** si nos centramos en mejorar solo el recall de la clase 1.

TRATAMIENTO DE LOS DATOS



Análisis del data set -EDA

Siendo no el foco de este trabajo, vamos a resumir en esta diapositiva todo lo referente a la naturaleza de los datos:

- Target desbalanceado.
- No hay nulos.
- Features Numéricas y Categóricas → Necesitarán transformaciones que se verán más adelante.

	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

Tipo de Variable	Columnas
Numéricas	Administrative
	Administrative_Duration
	Informational
	Informational_Duration
	ProductRelated
	ProductRelated_Duration
	BounceRates
	ExitRates
	PageValues
	Browser
	TrafficType
Categóricas	SpecialDay
	Month
	OperatingSystems
	Region
	VisitorType
	Weekend

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

Reducción de la dimensionalidad

- Como es sabido: **no siempre más es mejor**.
- Por tal motivo se van a estudiar todas las features mediante diferentes técnicas, para obtener aquellas que realmente aporten información al modelo y no introduzcan ruido y coste computacional innecesario.

1. Test de hipótesis:

- a) Numéricas → U de Mann-Whitney
- b) Categóricas → Chi₂

2. Feature Importance mediante modelo → XGBoost:

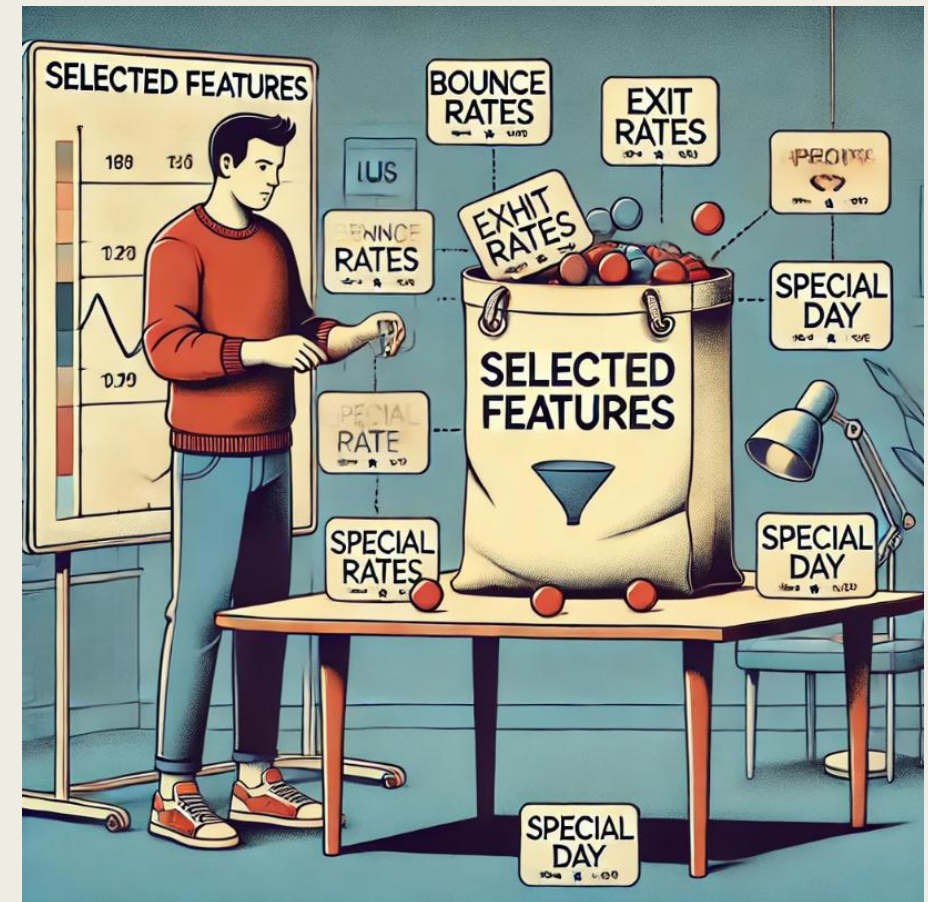
- a) aplicando scale_pos_weight
- b) aplicando under_sampling
- c) aplicando SMOTE

3. ANOVA y Mutual Information

4. SFS

5. RFE

6. Hard-Voting → Features finales



Feature importance - Test Hipótesis

Categorías

- Tabla de contingencia → `pd.crosstab`
- Test de hipótesis → `chi2_contingency`



	Chi2	p-value	Relación estadística
Month	384.934762	0.0	SI
VisitorType	135.251923	0.0	SI
SpecialDay	96.076906	0.0	SI
OperatingSystems	75.027056	0.0	SI
Weekend	10.390978	0.001266	SI
Region	9.252751	0.321425	NO



- “Region” no presenta una relación estadística, por lo que no será seleccionada.
- “Weekend” presenta una relación estadística baja, dejaremos que otros métodos la descarten.

Numéricas

- Creación de dos grupos
- Test de hipótesis → `mannwhitneyu`



	U-statistic	p-value	Relación estadística
Administrative	7421135.5	0.0	SI
Administrative_Duration	7487115.0	0.0	SI
Informational	8648742.0	0.0	SI
Informational_Duration	8711649.0	0.0	SI
ProductRelated	6792127.0	0.0	SI
ProductRelated_Duration	6502463.0	0.0	SI
BounceRates	12198493.0	0.0	SI
ExitRates	13981307.0	0.0	SI
PageValues	2718419.0	0.0	SI
Browser	9723556.5	0.071462	NO
TrafficType	9961029.0	0.894955	NO



- “Browser” no presenta relación estadística.
- “TrafficType” no presenta relación estadística.

Se crea las listas `features_hipotesis_num` y `features_hipotesis_cat`

Feature importance - Modelo XGBoost

Scale_pos_weight

```
pesos = y.value_counts()
peso_clases = pesos.iloc[0]/pesos.iloc[1]
modelo_xgb = ... scale_pos_weight=peso_clases...
SelectFromModel(estimator = modelo_xgb,
threshold= "median")
```



	Feature	Importancia
14	PageValues	0.416915
1	Month	0.105738
4	VisitorType	0.039078
0	SpecialDay	0.037045
11	ProductRelated_Duration	0.036667
10	ProductRelated	0.035024
6	Administrative	0.035011
12	BounceRates	0.033980
13	ExitRates	0.032777
15	Browser	0.030978
7	Administrative_Duration	0.030082
16	TrafficType	0.029815
9	Informational_Duration	0.028642
8	Informational	0.028235
5	Weekend	0.027769
2	OperatingSystems	0.026937
3	Region	0.025306

```
['SpecialDay',
'Month',
'VisitorType',
'Administrative',
'ProductRelated',
'ProductRelated_Duration',
'BounceRates',
'ExitRates',
'PageValues']
```

Under_sampling

```
RandomUnderSampler
xgb.XGBClassifier
modelo_xgb_us.feature_importances_
_
```



	Feature	Importancia
14	PageValues	0.296007
1	Month	0.156655
0	SpecialDay	0.103687
4	VisitorType	0.044468
5	Weekend	0.041797
10	ProductRelated	0.034665
6	Administrative	0.032663
13	ExitRates	0.031523
16	TrafficType	0.031279
12	BounceRates	0.030548
9	Informational_Duration	0.030417
11	ProductRelated_Duration	0.030118
8	Informational	0.029510
7	Administrative_Duration	0.029049
2	OperatingSystems	0.027863
15	Browser	0.025247
3	Region	0.024504

```
['SpecialDay',
'Month',
'VisitorType',
'Administrative',
'ProductRelated',
'ProductRelated_Duration',
'BounceRates',
'ExitRates',
'PageValues']
```

SMOTE

```
SMOTE
xgb.XGBClassifier
modelo_xgb_smote.feature_importances_
_
```



	Feature	Importancia
14	PageValues	0.401435
4	VisitorType	0.089001
1	Month	0.086630
5	Weekend	0.048332
2	OperatingSystems	0.047487
0	SpecialDay	0.044532
8	Informational	0.035589
15	Browser	0.028877
6	Administrative	0.028598
7	Administrative_Duration	0.028459
11	ProductRelated_Duration	0.027289
12	BounceRates	0.024467
10	ProductRelated	0.024195
13	ExitRates	0.022651
9	Informational_Duration	0.021952
16	TrafficType	0.020824
3	Region	0.019683

```
['SpecialDay',
'Month',
'VisitorType',
'Administrative',
'ProductRelated',
'BounceRates',
'ExitRates',
'PageValues',
'TrafficType']
```

Se crean sus respectivas listas

Feature importance mediante técnicas

ANOVA y Mutual Information

```
SelectKBest(f_classif, k=6)  
mutual_info_score(X_train_MI[col], y_train_MI)
```



```
['Administrative',  
'Administrative_Duration',  
'Informational',  
'Informational_Duration',  
'ProductRelated',  
'ProductRelated_Duration',
```



```
SpecialDay      0.196077  
Month           0.502692  
OperatingSystems 0.111523  
Region          0.009898  
VisitorType     0.159678  
Weekend         0.020133  
dtype: float64
```

SFS

```
SequentialFeatureSelector(modelo_x  
gb, n_features_to_select = 10...
```



```
['SpecialDay',  
'Month',  
'OperatingSystems',  
'VisitorType',  
'Weekend',  
'Informational',  
'ProductRelated',  
'ProductRelated_Duration',  
'PageValues',  
'Browser']
```

RFE

```
RFE(estimator = modelo_xgb,  
n_features_to_select= 10...
```



```
['SpecialDay',  
'Month',  
'VisitorType',  
'Administrative',  
'ProductRelated',  
'ProductRelated_Duration',  
'BounceRates',  
'ExitRates',  
'PageValues',  
'TrafficType']
```

Se crean sus respectivas listas

Feature importance - Hard Voting

Se aplica la función Counter y se contabilizan las repeticiones de cada feature en todas las litas



```
Counter({'SpecialDay': 7,  
        'Month': 7,  
        'VisitorType': 7,  
        'ProductRelated': 7,  
        'PageValues': 7,  
        'Administrative': 6,  
        'ProductRelated_Duration': 6,  
        'BounceRates': 6,  
        'ExitRates': 6,  
        'OperatingSystems': 3,  
        'Informational': 3,  
        'Administrative_Duration': 2,  
        'Informational_Duration': 2,  
        'Browser': 2,  
        'Weekend': 2,  
        'TrafficType': 2,  
        'Region': 1})
```



Se observa un gran salto entre las que tiene 7 y 6 repeticiones, respecto al resto.
Estas serán las features que se usarán para entrenar el modelo.



```
['SpecialDay',  
 'Month',  
 'PageValues',  
 'ProductRelated',  
 'VisitorType',  
 'Administrative',  
 'ProductRelated_Duration',  
 'BounceRates',  
 'ExitRates']
```

ANÁLISIS IMPACTO

FEATURE REDUCTION & SMOTE
VS
BASELINE

Baseline

Se va a crear un modelo baseline con XGBoost

Se comparará este baseline con otros 3 modelos, para validar si las técnicas usadas mejoran el recall medio:

1. Baseline con Feature reduction
2. Baseline con SMOTE
3. Baseline con Feature reduction + SMOTE

```
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)
```

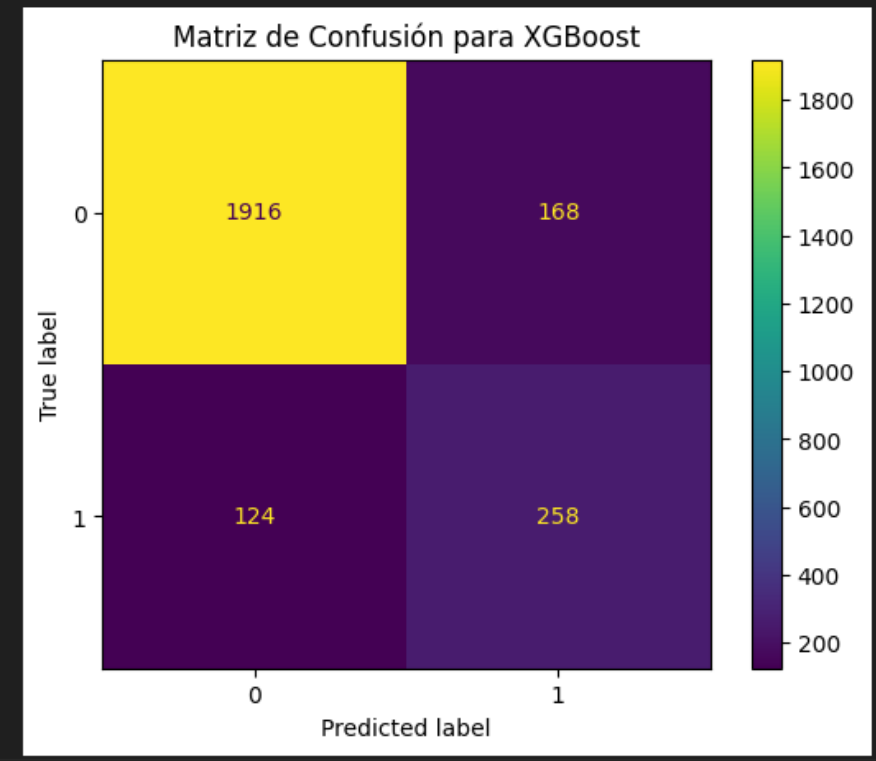
```
y_pred = modelo_xgb.predict(X_test)  
  
# Obtener el classification report  
print("Classification Report:")  
print(classification_report(y_test, y_pred))  
  
# Mostrar la matriz de confusión  
print("Confusion Matrix:")  
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)  
plt.title("Matriz de Confusión para XGBoost")  
plt.show()
```

Baseline

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:



Baseline con técnicas

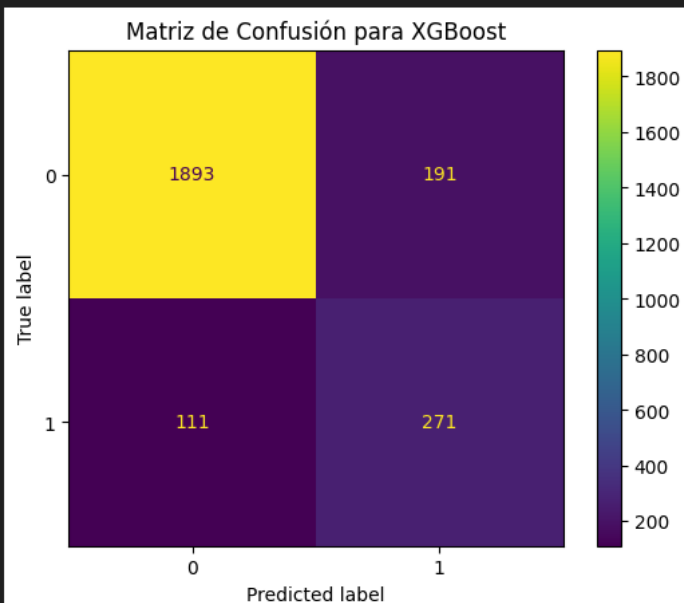
Baseline

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

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

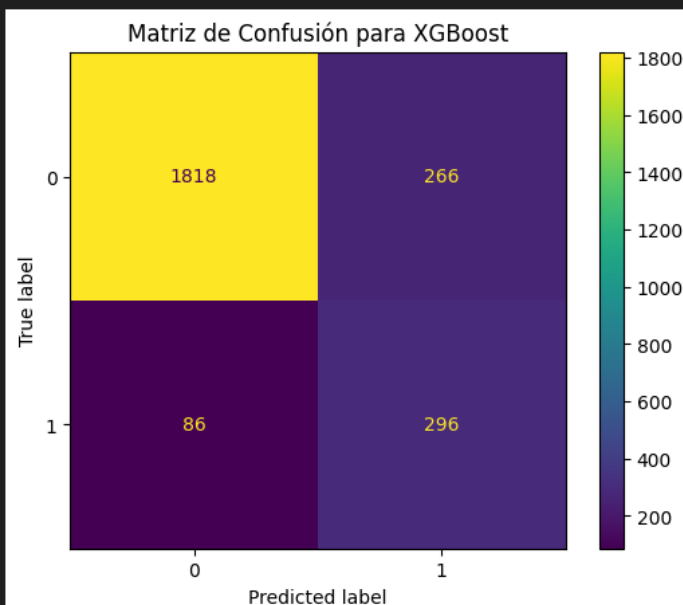
Confusion Matrix:



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

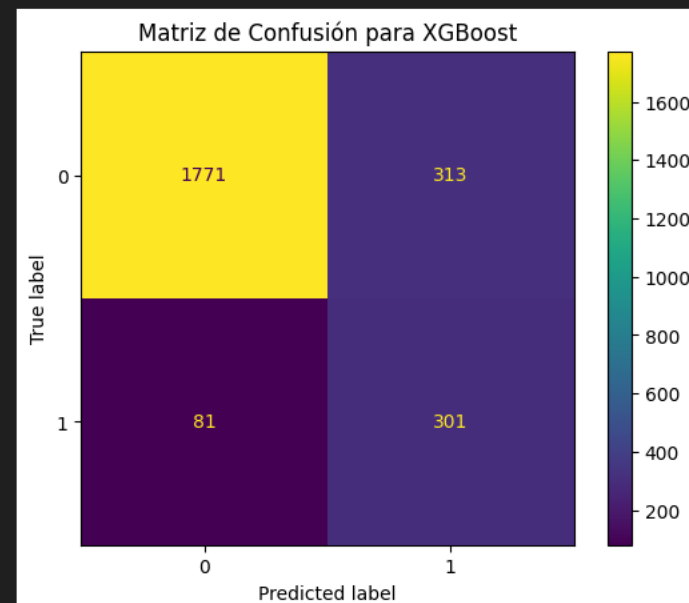
Confusion Matrix:



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:



Conclusiones Baseline

Feature Reduction vs Baseline:

1. La reducción de características no empeora el rendimiento, **mismo F1-Score**.
2. Mejora el **recall de la clase 1** (de 0.68 a 0.71) → **REDUCCIÓN DE RUIDO** → Detecta más compradores potenciales.

Baseline con SMOTE vs Baseline:

1. SMOTE mejora el **recall de la clase 1** (de 0.68 a 0.77) → **REDUCCIÓN DE SESGO**.
2. Este método es **más efectivo que la reducción de características** por sí sola (recall C1 de 0,71 a 0,77), ya que aumenta el número de compradores potenciales correctamente detectados.

Feature Reduction + SMOTE:

1. La combinación de ambas técnicas (reducción y SMOTE) obtiene el mejor **recall para la clase 1** (0.79).
2. Ambas técnicas maximizan la capacidad del modelo de identificar compradores potenciales.

Guardado variables relevantes

Guardado de variables relevantes para ser usadas durante el entrenamiento del modelo

Debido a que se quieren comparar los datos del base line con el modelo XGBoost tras aplicarle el grid_search, se han guardado, entre otras, las variables X_train, y_train, X_test e y_test, pues se quería conservar el mismo reparto de instancias en cada una de ellas.

```
# Carpeta de destino
variable_folder = "../features"
os.makedirs(variable_folder, exist_ok=True)

# Ruta del archivo guardado
variables_filename = os.path.join(variable_folder, "model_df_and_features.pkl")

# Guardar las variables
variables_to_save = {'df_shopping': df_shopping,
                    'fea_num_model': fea_num_model,
                    'fea_cat_model': fea_cat_model,
                    'features_to_drop': features_to_drop,
                    'target': target,
                    'X_train': X_train,
                    'y_train': y_train,
                    'X_test': X_test,
                    'y_test': y_test
                    }
joblib.dump(variables_to_save, variables_filename)

print(f"Variables guardadas como: {variables_filename}")
```

MODELOS MACHINE LEARNING



Pipeline

1 – Carga de datos

```
# Especificar la ruta al archivo donde se guardaron las variables
variable_folder = "../features"
variables_filename = os.path.join(variable_folder, "model_df_and_features.pkl")

# Cargar las variables
loaded_variables = joblib.load(variables_filename)

# Asignar las variables a los nombres originales
df_shopping = loaded_variables['df_shopping']
target = loaded_variables['target']
fea_num_model = loaded_variables['fea_num_model']
fea_cat_model = loaded_variables['fea_cat_model']
features_to_drop = loaded_variables['features_to_drop']
X_train = loaded_variables['X_train']
y_train = loaded_variables['y_train']
X_test = loaded_variables['X_test']
y_test = loaded_variables['y_test']
```

2 – Pipeline

```
# Transformadores para datos categóricos y numéricos
cat_transformer = Pipeline(steps=[
    ("onehot", OneHotEncoder(drop="first", handle_unknown='ignore'))
])

num_transformer = Pipeline(steps=[
    ("power_transformer", PowerTransformer(method='yeo-johnson', standardize=True))
])

# Preprocesador combinado
preprocessor = ColumnTransformer(
    transformers=[
        ("exclude", "drop", features_to_drop),
        ("num", num_transformer, fea_num_model),
        ("cat", cat_transformer, fea_cat_model)
    ], remainder='passthrough'
)

# Pipeline sin modelo
pipeline_imb_visualizacion = ImbPipeline(steps=[
    ("preprocessor", preprocessor),
    ("smote", SMOTE(random_state=42))
])
```

Visualización transformaciones

```
X_pipeline, y_pipeline = pipeline_imb_visualizacion.fit_resample(X_train, y_train)
```

	num_Administrative	num_ProductRelated	num_PageValues	num_ProductRelated_Duration	num_BounceRates	num_ExitRates
0	1.450438	0.593733	1.927643	0.592626	-0.285981	-0.826352
1	-0.991485	-1.130270	-0.530566	-0.697177	1.488583	1.562432
2	-0.991485	-1.130270	-0.530566	-1.220568	-0.797632	0.716121
3	0.562971	-0.003214	1.955261	0.509738	0.297545	0.321795
4	-0.991485	0.540581	1.980610	0.241800	-0.797632	-1.732567
...
16671	-0.991485	-1.162172	-0.530566	-1.020514	-0.797632	1.486310
16672	1.179061	-0.108096	-0.530566	0.949913	0.374801	0.320489
16673	1.374622	1.971934	1.944968	1.605791	-0.790427	-1.221718
16674	-0.991485	0.105189	-0.530566	-0.100993	0.984728	0.623353
16675	1.547537	1.096305	1.963587	1.026392	-0.156041	-0.396098

16676 rows x 23 columns

cat_Month_Oct	cat_Month_Sep	cat_SpecialDay_0.2	cat_SpecialDay_0.4	cat_SpecialDay_0.6	cat_SpecialDay_0.8
0.0	0.0	0.0	0.0	0.0	1.0
1.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	1.0
0.0	0.0	0.0	0.0	0.0	0.0
...
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0



```
X_train/y_train --> Transformación + SMOTE
X_test_trans --> Transformación
y_test
```

Modelos - Cross validation

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

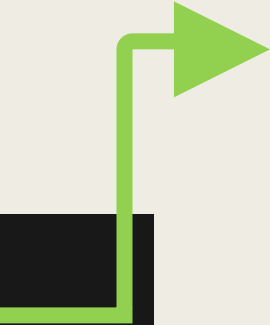
```
balanced_accuracy_scores = {}

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

Modelos - Grid Search

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")
```

Modelos - Evaluación

7 – Evaluación

```
# Modelos
model_names = [f"{model_name}_best_model" for model_name in models.keys()]

# Best params de los modelos
base_path = "../model/Best_params_model/Scoring_balanced_accuracy/"

for model_name in model_names:

    model_path = f"{base_path}{model_name}.pkl"
    modelo_entrenado = joblib.load(model_path)

    y_pred = modelo_entrenado.predict(X_test)

    # Balanced Accuracy y Classification Report
    balanced_acc = balanced_accuracy_score(y_test, y_pred)
    classification_rep = classification_report(y_test, y_pred)

    print(f"Resultados para el modelo: {model_name}")
    print(f"Balanced Accuracy en el conjunto de prueba: {balanced_acc:.4f}")
    print("\nReporte de Clasificación en el conjunto de prueba:")
    print(classification_rep)
    print("-" * 60)

    # Matriz de confusión
    ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
    plt.title(f"Confusion Matrix for {model_name} on Test Data")
    plt.show()
```

Baseline

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

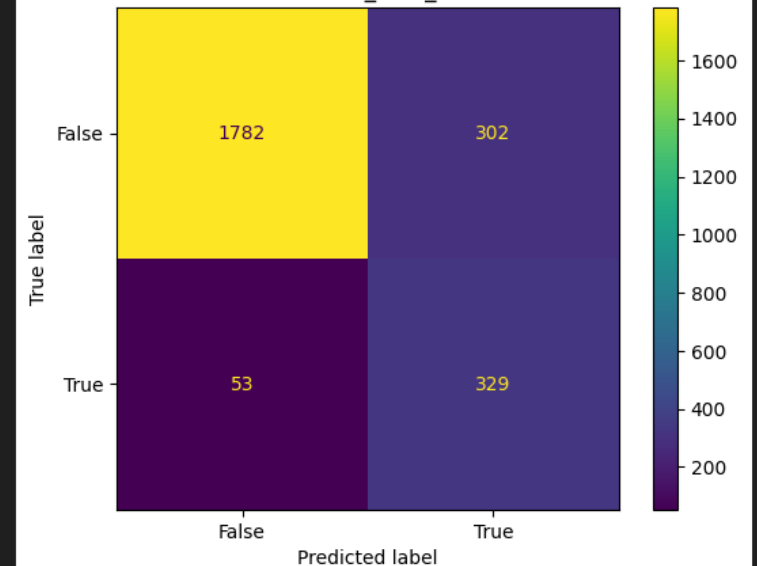
Mejor modelo

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

Confusion Matrix for SVM_best_model on Test Data



DEEP LEARNING



Deep Learning - Transformación

1 - Carga de datos

```
X_y_shopping_trans = pd.read_csv("../data/processed/shopping_transformado_X_train_y_train_only.csv")

variable_folder = "../data/processed"
variables_filename = os.path.join(variable_folder, "set_datos_transformados_SMOTE.pkl")

loaded_variables = joblib.load(variables_filename)

df_shopping_transformado = loaded_variables['shopping_trans_X_y']
X_test = loaded_variables['X_test_trans']
y_test = loaded_variables['y_test']
```

```
target = "Revenue"
```

```
X_train = X_y_shopping_trans.drop(columns=target, axis=1)
y_train = X_y_shopping_trans[target]
```

2 - Transformación datos

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



Aunque los valores iniciales están muy próximos a 0 (entre -2 y 2), se va a proceder a estandarizarlos entre -1 y 1 para un mejor desempeño de la red neuronal

Deep Learning – Red neuronal

1 – Modelo

```
model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    BatchNormalization(),
    Dropout(0.3),
    Dense(64, activation='relu'),
    BatchNormalization(),
    Dropout(0.3),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')
])
```



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

2 – Compilación

```
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
    loss='binary_crossentropy',
    metrics=['accuracy', tf.keras.metrics.Recall(name='recall')]
)
```

3 – Entrenamiento

```
early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=100,
    restore_best_weights=True
)
```



Si, patience esta a 100, queremos que haga todas las épocas.

Deep Learning - Entrenamiento

3 - Entrenamiento

```
early_stopping = tf.keras.callbacks.EarlyStopping(  
    monitor='val_loss',  
    patience=100,  
    restore_best_weights=True  
)
```

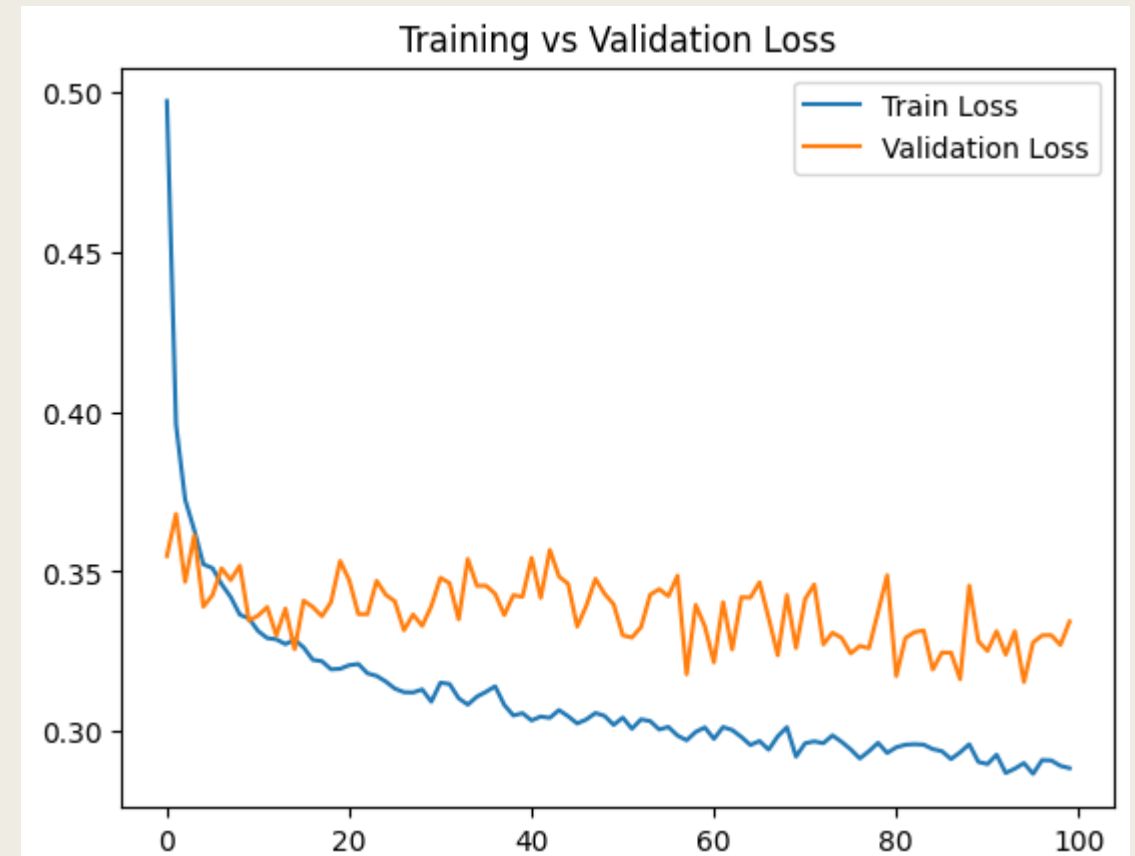
```
history = model.fit(  
    X_train_scaled, y_train,  
    validation_split=0.2,  
    batch_size=16,  
    epochs=100,  
    callbacks=[early_stopping],  
    verbose=2  
)
```

Se ve claramente como llegar a 100 épocas no tiene sentido.
La pérdida en validación incluso sube en muchos puntos.

El modelo está memorizando datos →
OVERFITTING



4 - Evolución función de pérdida



Deep Learning - Entrenamiento

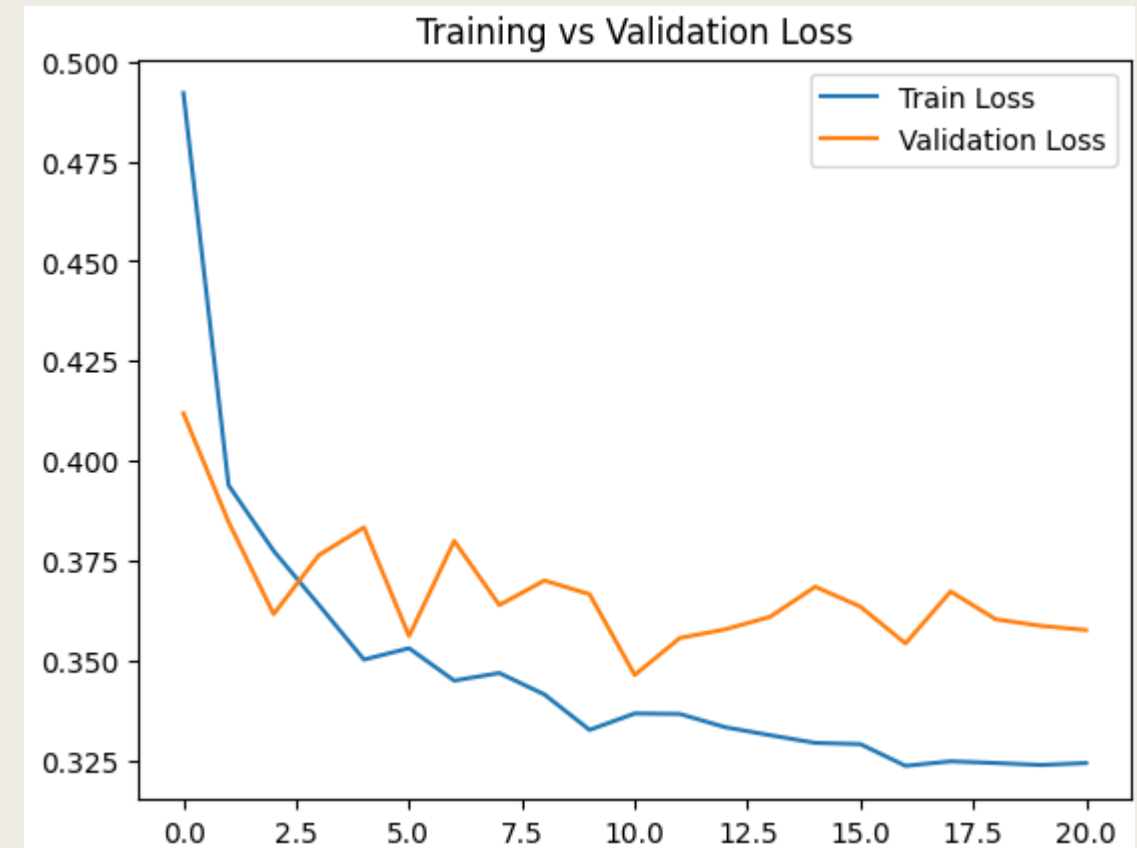
3.a - Entrenamiento

```
early_stopping_2 = tf.keras.callbacks.EarlyStopping(  
    monitor='val_loss',  
    patience=10,  
    restore_best_weights=True  
)  
  
history_2 = model_2.fit(  
    X_train_scaled, y_train,  
    validation_split=0.2,  
    batch_size=16,  
    epochs=100,  
    callbacks=[early_stopping_2],  
    verbose=2  
)
```

En la época 11 ya se alcanza el valor mínimo de la función de coste.
Tras 10 épocas más, no se consigue mejorar.
El modelo habrá generalizado mejor.



4.a - Evolución función de pérdida



100 épocas

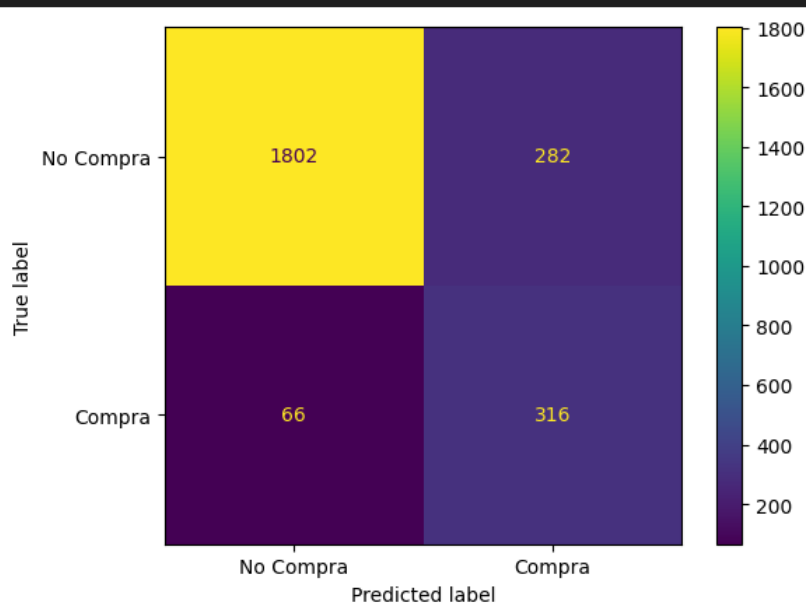
ML - SVM

11 épocas

Loss: 0.2913, Accuracy: 0.8589, Recall: 0.8272
78/78 — 0s 701us/step

Classification Report:

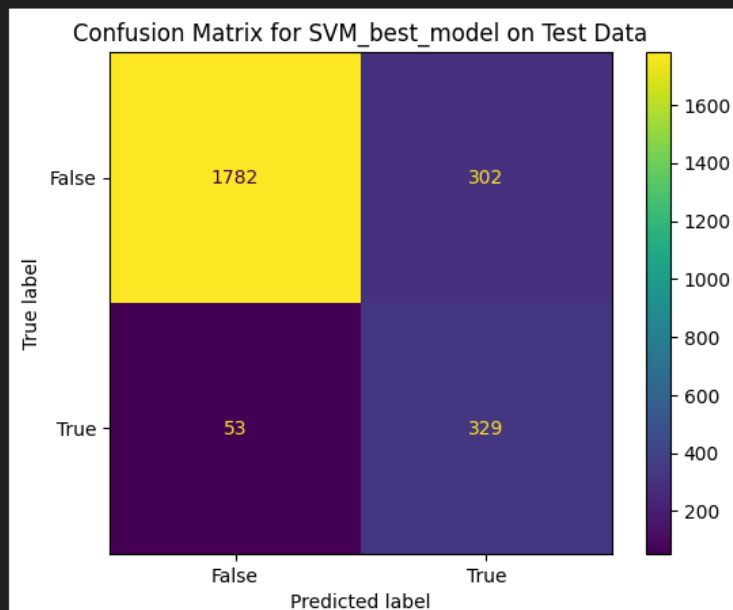
	precision	recall	f1-score	support
No Compra	0.96	0.86	0.91	2084
Compra	0.53	0.83	0.64	382
accuracy			0.86	2466
macro avg	0.75	0.85	0.78	2466
weighted avg	0.90	0.86	0.87	2466



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

Reporte de Clasificación en el conjunto de prueba:

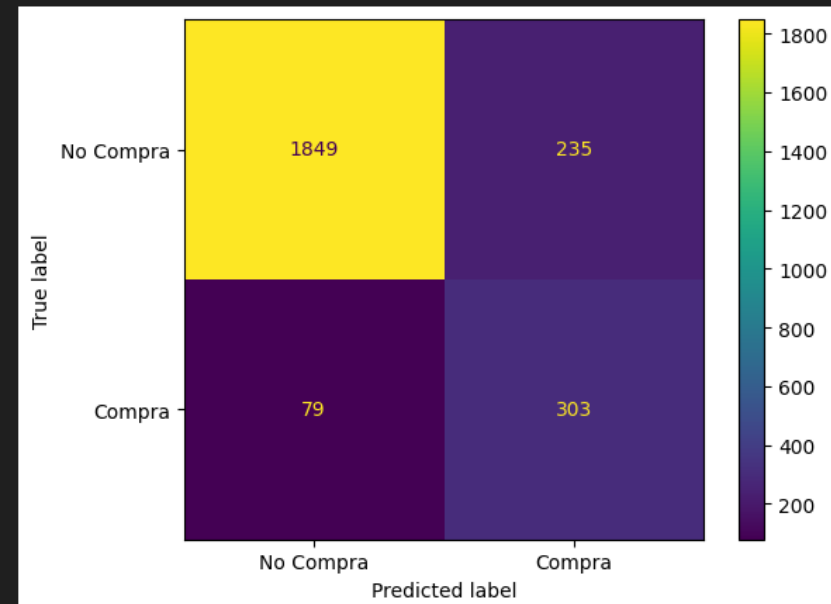
	precision	recall	f1-score	support
False	0.97	0.86	0.91	2084
True	0.52	0.86	0.65	382
accuracy			0.86	2466
macro avg	0.75	0.86	0.78	2466
weighted avg	0.90	0.86	0.87	2466



Loss: 0.2956, Accuracy: 0.8727, Recall: 0.7932
78/78 — 0s 721us/step

Classification Report:

	precision	recall	f1-score	support
No Compra	0.96	0.89	0.92	2084
Compra	0.56	0.79	0.66	382
accuracy			0.87	2466
macro avg	0.76	0.84	0.79	2466
weighted avg	0.90	0.87	0.88	2466



BEST MODEL
SVM - PROBABILIDAD

SVM - Probabilidades clases

- Nuestro amigo Manuel nos pidió si podía saber en tiempo real si alguien iba a comprar o no, para poder lanzar campañas de marketing personalizado
- Para poder realizar esta tarea, es necesario conocer las probabilidades de cada una de las clases.
- Para ello, vamos a entrenar de nuevo el mejor modelo, pero esta vez, con el argumento siguiente:
 - Probability = True

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

Gracias a esto, el modelo podrá ser el input de entrada de otros mecanismos, que permitan la emisión de un tipo de promoción, según el resultado de la probabilidad en cada clase.

SVM - Probabilidades clases

- El modelo se ha entrenado de la misma manera que se ha mostrado anteriormente, por lo que omitiremos esa información.
- Se muestra únicamente la parte del calculo de probabilidades.

```
1 # Hacer predicciones con probabilidades del mejor modelo
2 y_pred_proba = best_model.predict_proba(X_test)
3 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]]
```

- De esta manera, se podría crear un sistema que, según rangos de probabilidades para cada clase, emitiría unas promociones diferentes, optimizando los recursos.

GRACIAS

