Universidad de Antioquia 2024-1

Resultados Caso de Estudio E-Corp

ESTEBAN CARDONA, GILBERTO GIL, LEIDYS GUERRERO, MATEO CAICEDO

Analítica para la toma de Decisiones

Introducción

E-Corp, una empresa especializada en productos de lujo, enfrenta el desafío de **maximizar las ventas en su sitio web de comercio electrónico**. Para abordar este problema, proponemos utilizar modelos de Aprendizaje Automático (ML) para optimizar la inversión en publicidad digital. Nuestro enfoque incluye análisis de datos, selección de características y entrenamiento de modelos de ML como **Regresión Logística, *Random Forest* y *Gradient Boosting*,** seguido de estrategias de segmentación para mejorar el rendimiento en línea y maximizar el retorno de la inversión en e-commerce de lujo.

Exploración de los Datos

Base de Datos Inicial

La base de datos inicial cuenta con un total de **12.330 registros** divididos en 18 columnas asociadas a nuestras variables de estudio.

Su registro se encuentra almacenado en un ***data frame*** denominado como `df_ventas`.

Se tiene entonces como variable objetivo (Y) a la variable ***'Purchase'*** la cual es una variable tipo ***'bool'*** o también puede ser llamada como una **variable dicotómica.**

```
#   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Reviews                  12330 non-null  int64
 1   Reviews_Duration         12330 non-null  float64
 2   Informational            12330 non-null  int64
 3   Informational_Duration   12330 non-null  float64
 4   ProductRelated           12330 non-null  int64
 5   ProductRelated_Duration  12330 non-null  float64
 6   BounceRates              12330 non-null  float64
 7   ExitRates                12330 non-null  float64
 8   PageValues               12330 non-null  float64
 9   SpecialDay               12330 non-null  float64
 10  Month                    12330 non-null  object
 11  OperatingSystems         12330 non-null  int64
 12  Browser                  12330 non-null  int64
 13  Region                   12330 non-null  int64
 14  TrafficType              12330 non-null  int64
 15  VisitorType              12330 non-null  object
 16  Weekend                  12330 non-null  bool
 17  Purchase                 12330 non-null  bool
```

Resultados en terminal

Exploración de los Datos

Cambios Iniciales

- Una de las principales conversiones que se consideran necesarias es convertir la variable ***'Month'*** a una categórica ordinal.

- Las variables asociadas a ***'Weekend'*** y ***'Purchase'*** vienen por defecto con valores de ***True*** y ***False***, y con el fin de trabajar todo el ***dataframe*** con valores numéricos, se realiza un reemplazo en los datos por 1 y 0 respectivamente.

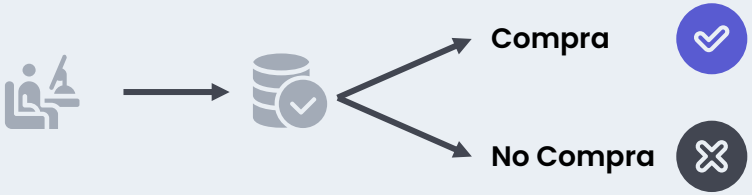
```
# Lista de meses
meses = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'June',
'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

# Crear un diccionario de mapeo de meses
mapeo_meses = {mes: idx + 1 for idx, mes in
enumerate(meses)}

# Mapear los meses en el DataFrame
df_ventas['Month'] = df_ventas['Month'].map(mapeo_meses)

df_ventas

# Reemplazar True con 1 y False con 0
df_ventas = df_ventas.replace({True: 1, False: 0})
```

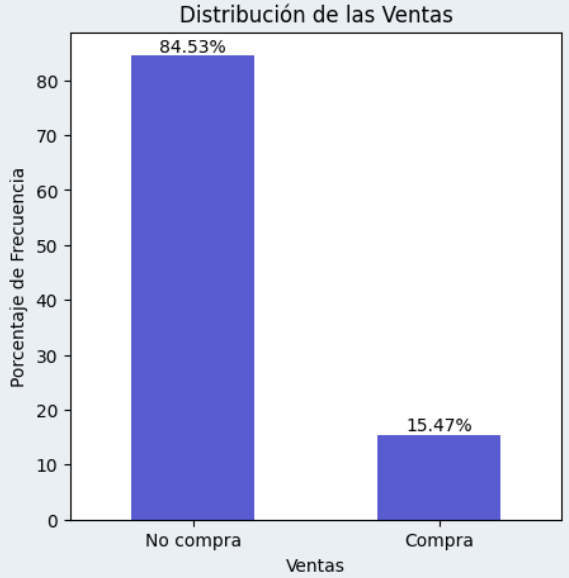
Resultados en terminal

Exploración de los Datos

Comportamiento de la Variable Objetivo

Nuestra variable objetivo ***'Purchase'*** básicamente se refleja como la acción de un usuario al realizar una compra efectiva, por ello, es importante ver cómo ha sido el comportamiento de las ventas en el tiempo de análisis.





Resultados en terminal

Análisis de Variables

Numéricas

Del conjunto de datos de 18 variables, 8 de ellas pertenecen a **variables numéricas**, reconocidas en el *dataframe* como:

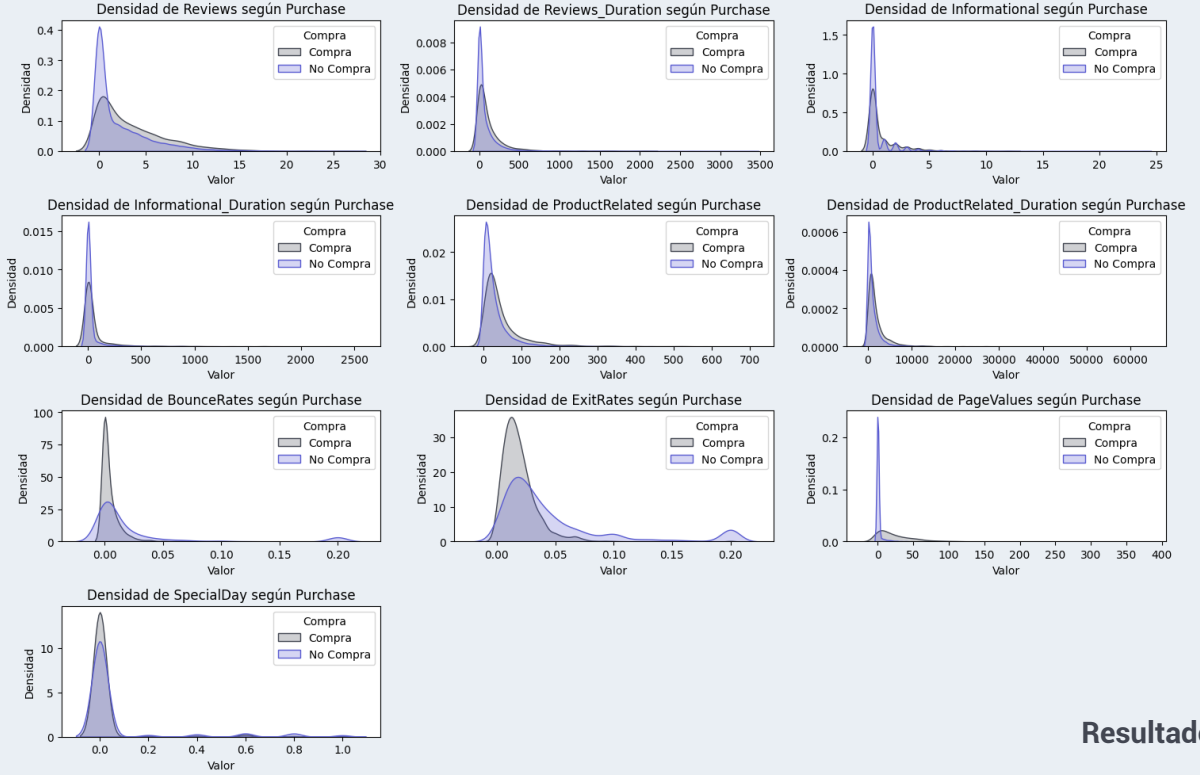
```
num_var = ['Reviews', 'Reviews_Duration',
'Informational',  'Informational_Duration',
'ProductRelated','ProductRelated_Duration',  'BounceRates',  'ExitRates',  'Pag
eValues', 'SpecialDay']
```

Categóricas

En el mismo sentido, el restante de variables pertenecen a variables categóricas, sumando 10 **variables categóricas**, reconocidas en el *dataframe* como:

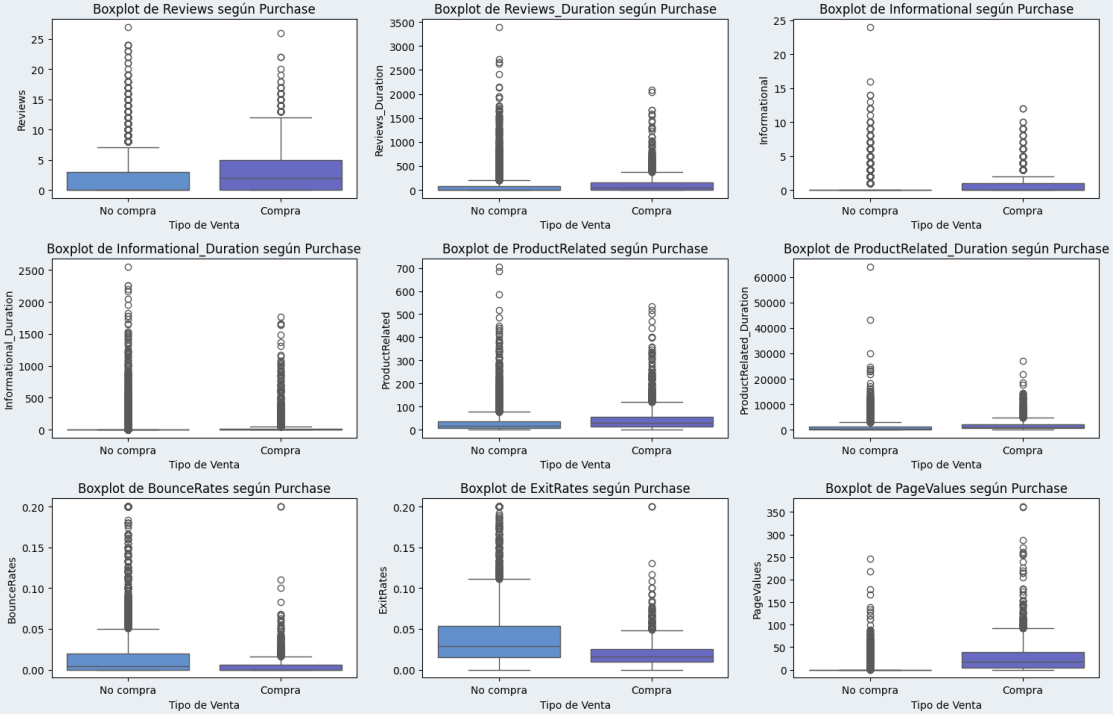
```
cat_var = ['Month', 'OperatingSystems', 'Browser', 'Region', 'TrafficType',
'VisitorType', 'Weekend', 'Purchase']
```

Variables Numéricas



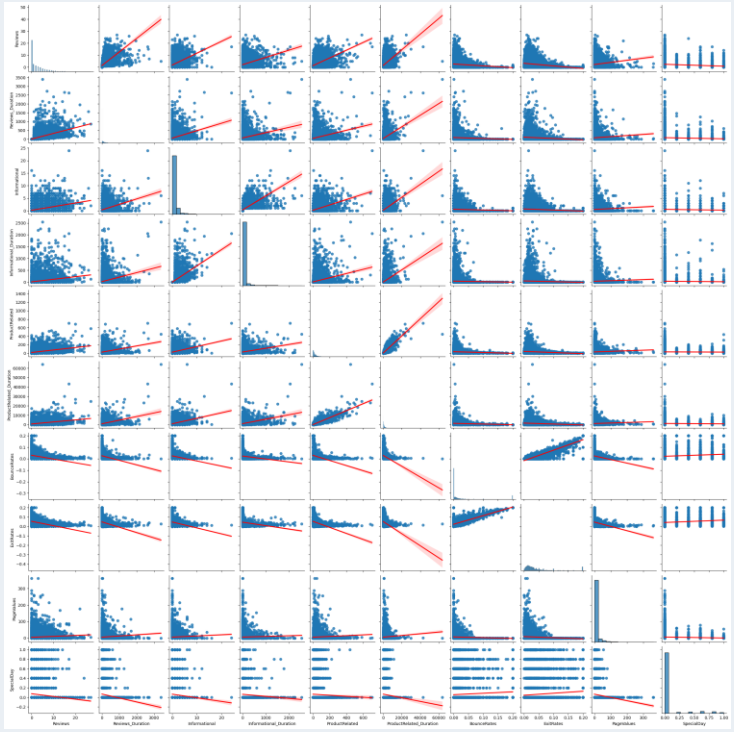
Resultados en terminal

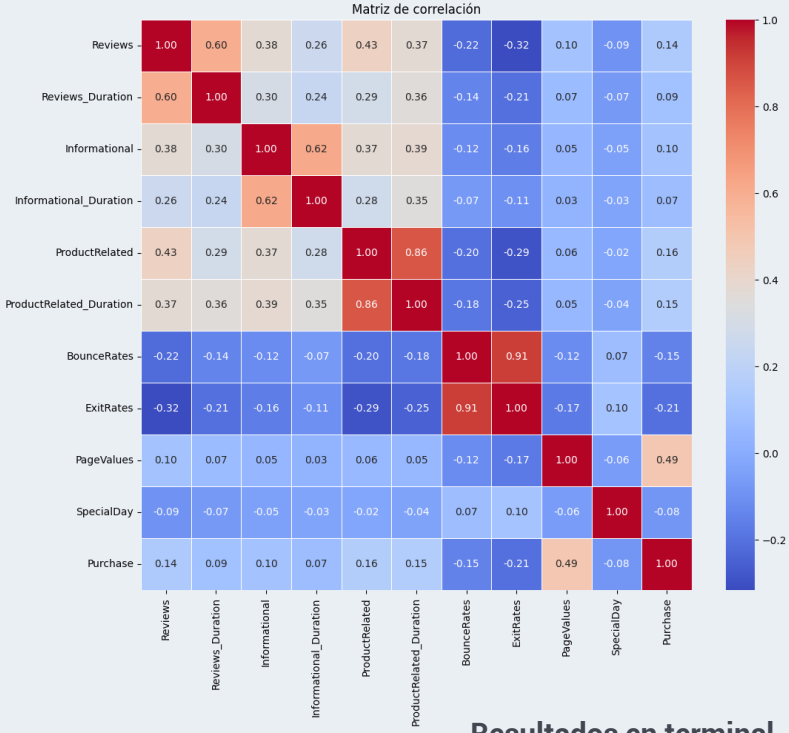
Variables Numéricas



Resultados en terminal

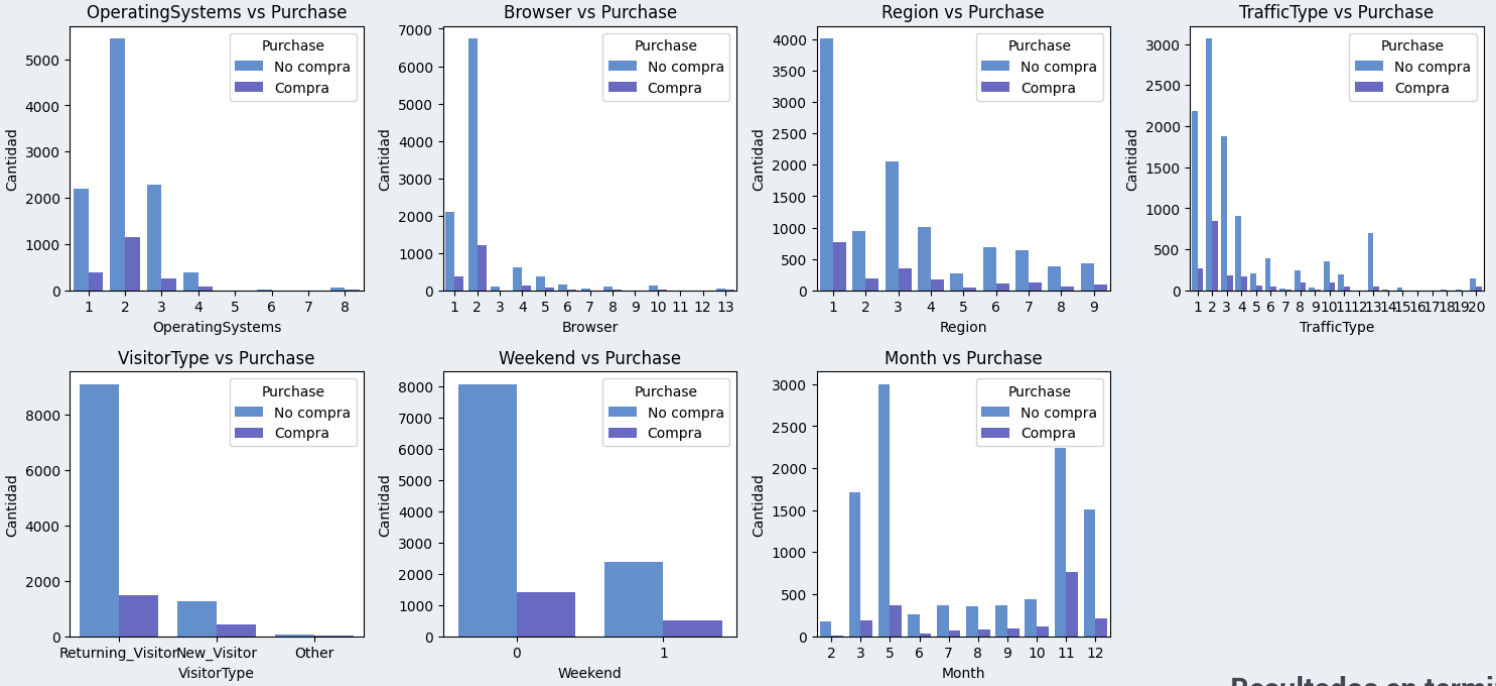
Variables Numéricas





Resultados en terminal

Variables Categóricas



Resultados en terminal

Variables Categóricas

Identificación de Variables

Para la segmentación de variables categóricas se utilizó una tabla de contingencia chi2 con el fin de identificar cuáles de ellas eran significativas para el modelo, de acuerdo con los resultados la única que no sería significativa es la Región asociada a su variable homónima ***'Region'.***

```
from scipy.stats import chi2_contingency
```

Variable	Chi2	P-value	Significativo
Month	384.934762	2.238786e-77	Sí
OperatingSystems	75.027056	1.416094e-13	Sí
Browser	27.715299	6.087543e-03	Sí
Region	9.252751	3.214250e-01	No
TrafficType	373.145565	1.652735e-67	Sí
VisitorType	135.251923	4.269904e-30	Sí
Weekend	10.390978	1.266325e-03	Sí
Purchase	12322.355847	0.000000e+00	Sí

Resultados en terminal

Actualización de Datos

Analítica para la toma de Decisiones

Dummizar y Estandarizar Variables

Conversión de Variables

La **dummyficación** es una técnica comúnmente empleada para codificar variables categóricas en variables binarias (0 o 1), lo que permite que sean utilizadas por algoritmos de aprendizaje automático.

Por otro lado, la estandarización de variables es un proceso común en el preprocesamiento de datos que se utiliza para asegurar que todas las variables tengan la misma escala. En este caso utilizado para trabajar con el modelo de **Regresión Logística.**

```
# Obtén variables dummy solo para las columnas especificadas
df_dummies = pd.get_dummies(df_ventas[cat_var1], columns=cat_var1)

# Combina las variables dummy con el DataFrame original
df_ventas1 = pd.concat([df_ventas.drop(cat_var1, axis=1),
df_dummies], axis=1)

# Añade la columna 'Purchase' del DataFrame original a df_ventas
df_ventas1['Purchase'] = df_ventas['Purchase']

from sklearn.preprocessing import MinMaxScaler

# Seleccionar las columnas que deseas estandarizar
columnas_a_estandarizar = ['Reviews', 'Reviews_Duration',
'Informational', 'Informational_Duration', 'ProductRelated',
'ProductRelated_Duration','BounceRates','ExitRates', 'PageValues']

# Crear el MinMaxScaler
scaler = MinMaxScaler()

# Separar las características (X) y la variable objetivo (y)
df_scaled = df_ventas1.copy()

# Aplicar el scaler solo a las columnas seleccionadas
df_scaled[columnas_a_estandarizar] =
scaler.fit_transform(df_scaled[columnas_a_estandarizar])
```

Resultados en terminal

Preparar los Datos

Analítica para la toma de Decisiones

Preparación de los Datos

Entrenamiento y Validación

Para la preparación de los datos, entonces, se utilizó una proporción **20%-80%** para los datos de entrenamiento y validación, respectivamente, contando con un total de:

- **Tamaño del conjunto de entrenamiento:**
  (9864, 65).
- **Tamaño del conjunto de validación:**
  (2466, 65).

```
# Separar las características (X) y la variable objetivo (y)
X = df_ventas1.drop(['Purchase'], axis=1)
y = df_ventas1['Purchase']

# Dividir el conjunto de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=28)
```

Resultados en terminal

Modelos Base

Analítica para la toma de Decisiones

Modelos de ML

Regresión Logística Base

```
Métricas de desempeño sobre el conjunto de entrenamiento:
Accuracy Score: 82.55%
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.84      0.89      8319
           1       0.47      0.76      0.58      1545

    accuracy                           0.83      9864
   macro avg       0.71      0.80      0.73      9864
weighted avg       0.87      0.83      0.84      9864


Métricas de desempeño sobre el conjunto de validación:
Accuracy Score: 81.79%
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.83      0.89      2103
           1       0.43      0.75      0.55       363

    accuracy                           0.82      2466
   macro avg       0.69      0.79      0.72      2466
weighted avg       0.87      0.82      0.84      2466
```

```python
# REGRESIÓN LOGÍSTICA BASE
logistic_model = LogisticRegression(random_state=28,
class_weight='balanced')

# Entrenar el modelo
logistic_model.fit(X_train, y_train)

# Predecir sobre los datos de entrenamiento y prueba
y_train_pred = logistic_model.predict(X_train)
y_test_pred = logistic_model.predict(X_test)

# Evaluar el modelo
eval_model(y_train_pred, y_test_pred)
```

Resultados en terminal

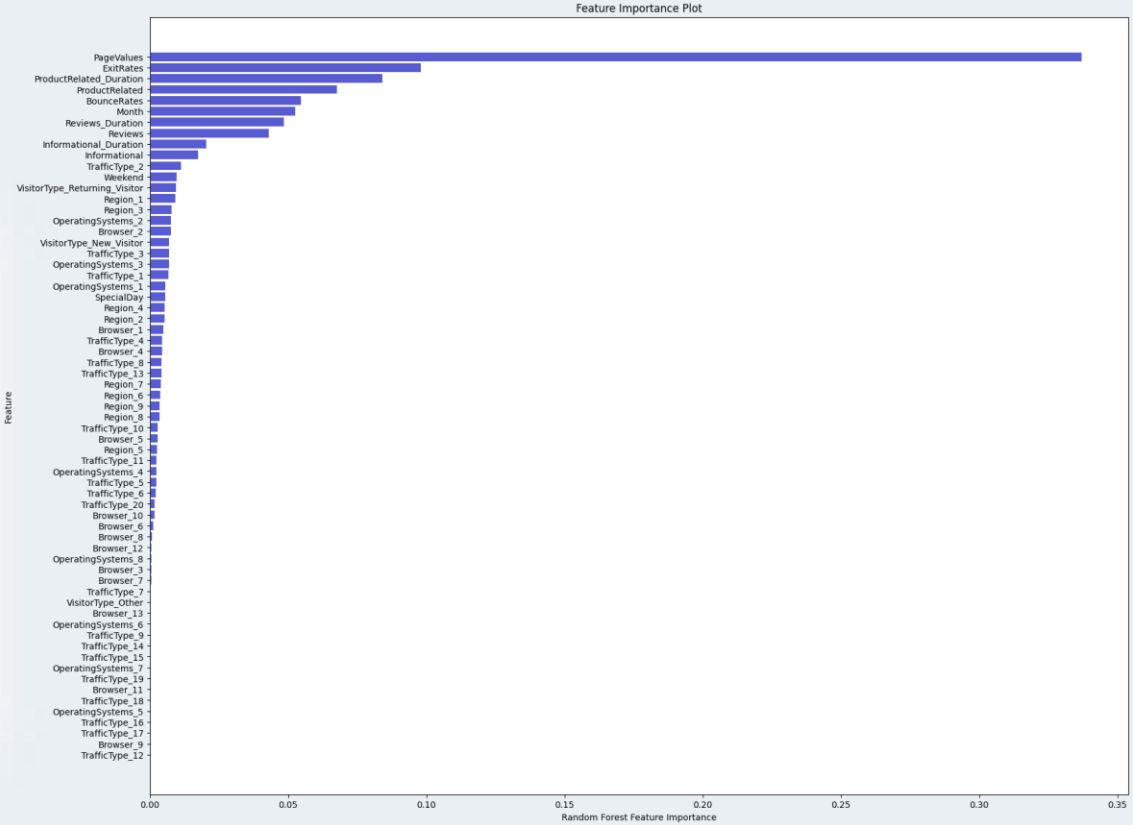
Modelos de ML

Random Forest Base

```
Métricas de desempeño sobre el conjunto de entrenamiento:
Accuracy Score: 100.00%
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      8319
           1       1.00      1.00      1.00      1545

    accuracy                           1.00      9864
   macro avg       1.00      1.00      1.00      9864
weighted avg       1.00      1.00      1.00      9864


Métricas de desempeño sobre el conjunto de validación:
Accuracy Score: 90.11%
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.97      0.94      2103
           1       0.75      0.49      0.59       363

    accuracy                           0.90      2466
   macro avg       0.83      0.73      0.77      2466
weighted avg       0.89      0.90      0.89      2466
```

```python
#RANDOM FOREST BASE
RanFor_base = RandomForestClassifier(class_weight='balanced',
random_state=28, n_jobs=-1)

RanFor_base.fit(X_train, y_train)

# Predicciones sobre el conjunto de entrenamiento y validación
y_train_base = RanFor_base.predict(X_train)
y_test_base = RanFor_base.predict(X_test)

# Evaluar el modelo
eval_model(y_train_base, y_test_base)
```

Resultados en terminal



Importancia de Características

El gráfico mostrado es un gráfico de barras horizontales que representa la importancia de las características (features) en un modelo de ***Random Forest.***

Resultados en terminal

Modelos de ML

G Boost Base

```
Métricas de desempeño sobre el conjunto de entrenamiento:
Accuracy Score: 91.68%
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.97      0.95      8319
           1       0.78      0.65      0.71      1545

    accuracy                           0.92      9864
   macro avg       0.86      0.81      0.83      9864
weighted avg       0.91      0.92      0.91      9864


Métricas de desempeño sobre el conjunto de validación:
Accuracy Score: 90.27%
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.96      0.94      2103
           1       0.70      0.60      0.64       363

    accuracy                           0.90      2466
   macro avg       0.82      0.78      0.79      2466
weighted avg       0.90      0.90      0.90      2466
```

```python
#RANDOM FOREST BASE
RanFor_base = RandomForestClassifier(class_weight='balanced',
random_state=28, n_jobs=-1)

RanFor_base.fit(X_train, y_train)

# Predicciones sobre el conjunto de entrenamiento y validación
y_train_base = RanFor_base.predict(X_train)
y_test_base = RanFor_base.predict(X_test)

# Evaluar el modelo
eval_model(y_train_base, y_test_base)
```

Resultados en terminal

Modelos de ML

XG Boost Base

```
Métricas de desempeño sobre el conjunto de entrenamiento:
Accuracy Score: 98.60%
Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      8319
           1       0.99      0.92      0.95      1545

    accuracy                           0.99      9864
   macro avg       0.99      0.96      0.97      9864
weighted avg       0.99      0.99      0.99      9864


Métricas de desempeño sobre el conjunto de validación:
Accuracy Score: 89.78%
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.95      0.94      2103
           1       0.68      0.57      0.62       363

    accuracy                           0.90      2466
   macro avg       0.81      0.76      0.78      2466
weighted avg       0.89      0.90      0.89      2466
```

```python
#GRADIENT BOOSTING XTREME

from xgboost import XGBClassifier

xgb = XGBClassifier(n_estimators = 100, random_state = 28)

xgb.fit(X_train, y_train)

# Predicciones sobre el conjunto de entrenamiento y validación
y_train_xgb = xgb.predict(X_train)
y_test_base_xgb = xgb.predict(X_test)

#Evaluar el modelo
eval_model(y_train_xgb, y_test_base_xgb)
```

Resultados en terminal

Selección de Características

Analítica para la toma de Decisiones

Filtrado, Wrapper y SelectFromModel

Variables Principales	SKB	RFE	SFM
Reviews	x	x	x
Reviews_Duration	x	x	x
Informational	x	x	x
ProductRelated	x	x	x
ProductRelated_Duration	x	x	x
BounceRates	x	x	x
ExitRates	x	x	x
PageValues	x	x	x
SpecialDay	x		
Month	x	x	x
OperatingSystems_3	x		
TrafficType_2	x	x	
TrafficType_3			x
VisitorType_New_Visitor			x
VisitorType_Returning_Visitor	x	x	x
Informational_Duration	x	x	
OperatingSystems_2		x	
Browser_2		x	
Region_1		x	

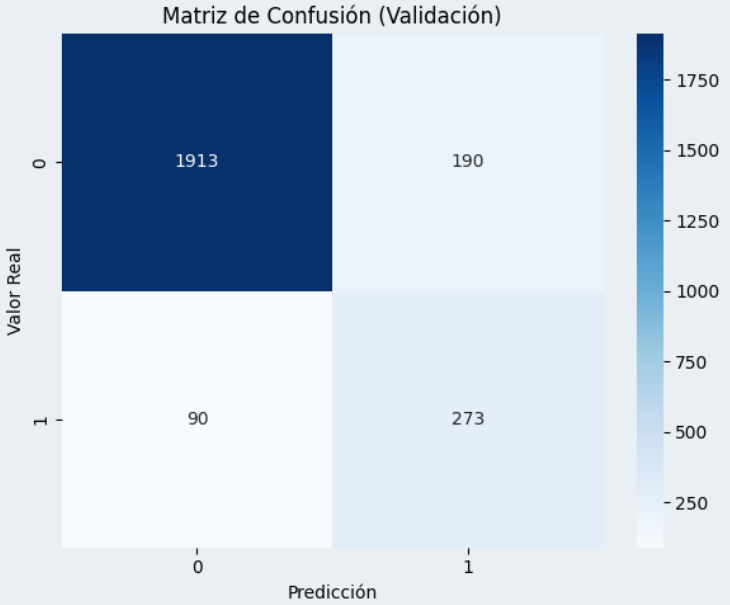
```
# Para clasificación
from sklearn.feature_selection import SelectKBest, f_classif

# Función de filtro de caracteristicas - stadis. scores
def select_kbest_classification(X,y,score_f,k):
    sel_kb = SelectKBest(score_func=score_f, k=k)
    sel_kb.fit(X,y)
    new_cols = sel_kb.get_support()
    #print("Scores:\n", sel_kb.scores_, "\nP-values:\n",
sel_kb.pvalues_)
    return new_cols

from sklearn.feature_selection import RFE

# Función recursiva de selección de características
def recursive_feature_selection(X,y,model,k):
  rfe = RFE(model, n_features_to_select=k, step=1)
  fit = rfe.fit(X, y)
  X_new = fit.support_
  print("Num Features: %s" % (fit.n_features_))
  print("Selected Features: %s" % (fit.support_))
  print("Feature Ranking: %s" % (fit.ranking_))

  return X_new

from sklearn.feature_selection import SelectFromModel
```

Resultados en terminal

Ajuste de Modelos

Analítica para la toma de Decisiones

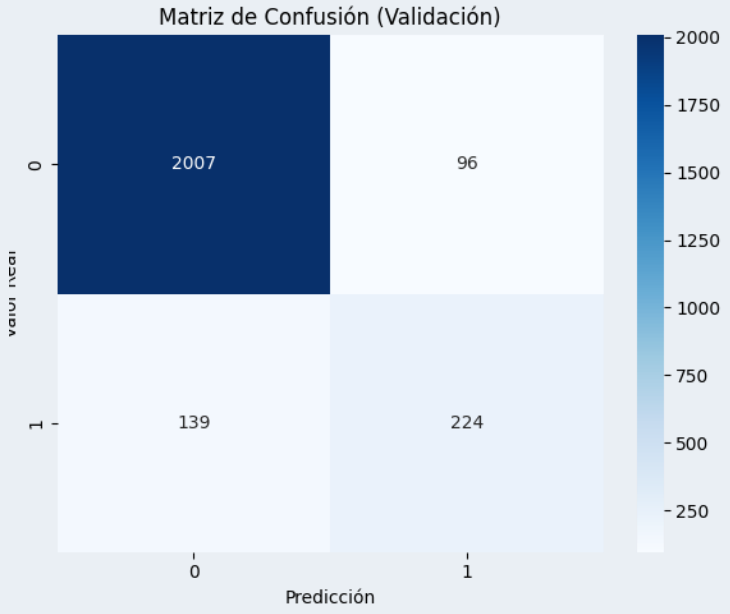
Random Forest Ajustado




```
from sklearn.model_selection import GridSearchCV

Métricas de desempeño sobre el conjunto de entrenamiento:
Accuracy Score: 93.63%
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.94      0.96      8319
           1       0.74      0.93      0.82      1545

    accuracy                           0.94      9864
   macro avg       0.86      0.93      0.89      9864
weighted avg       0.95      0.94      0.94      9864


Métricas de desempeño sobre el conjunto de validación:
Accuracy Score: 88.65%
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.91      0.93      2103
           1       0.59      0.75      0.66       363

    accuracy                           0.89      2466
   macro avg       0.77      0.83      0.80      2466
weighted avg       0.90      0.89      0.89      2466
```

Resultados en terminal

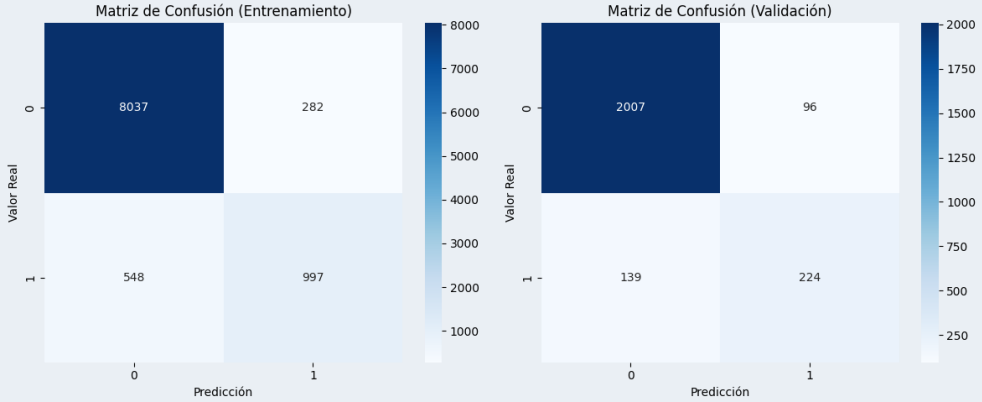
XGBoost Ajustado



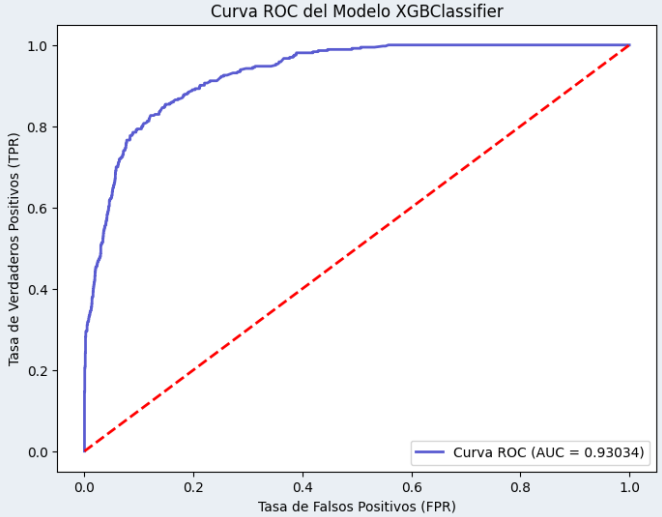

```
from sklearn.model_selection import GridSearchCV

Métricas de desempeño sobre el conjunto de entrenamiento:
Accuracy Score: 91.59%
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.97      0.95      8319
           1       0.78      0.65      0.71      1545

    accuracy                           0.92      9864
   macro avg       0.86      0.81      0.83      9864
weighted avg       0.91      0.92      0.91      9864


Métricas de desempeño sobre el conjunto de validación:
Accuracy Score: 90.47%
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.95      0.94      2103
           1       0.70      0.62      0.66       363

    accuracy                           0.90      2466
   macro avg       0.82      0.79      0.80      2466
weighted avg       0.90      0.90      0.90      2466
```

Resultados en terminal

XGBoost Ajustado

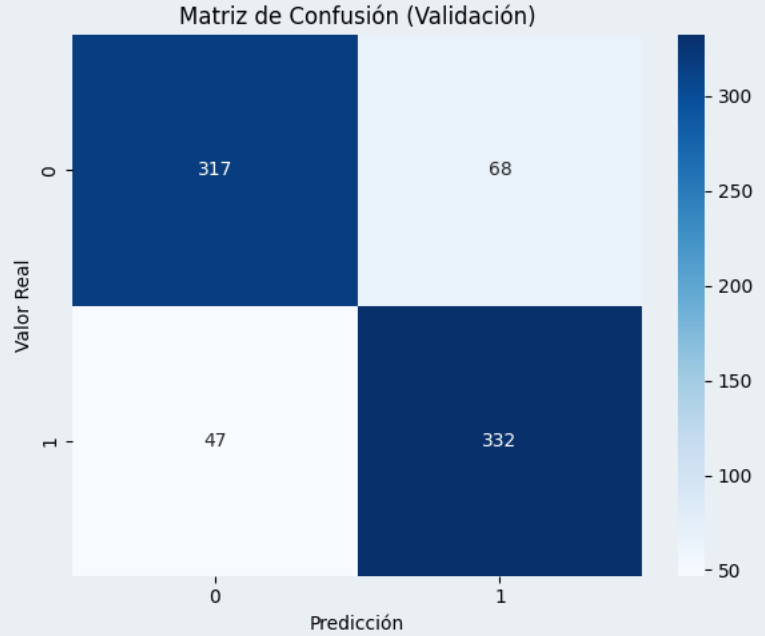



```
from sklearn.metrics import roc_curve, auc
```



Resultados en terminal

XGBoost Ajustado Submuestreo




```
from imblearn.under_sampling import RandomUnderSampler
from collections import Counter

Métricas de desempeño sobre el conjunto de entrenamiento:
Accuracy Score: 86.63%
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.88      0.87      1523
           1       0.87      0.86      0.87      1529

    accuracy                           0.87      3052
   macro avg       0.87      0.87      0.87      3052
weighted avg       0.87      0.87      0.87      3052


Métricas de desempeño sobre el conjunto de validación:
Accuracy Score: 84.95%
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.82      0.85       385
           1       0.83      0.88      0.85       379

    accuracy                           0.85       764
   macro avg       0.85      0.85      0.85       764
weighted avg       0.85      0.85      0.85       764
```

Resultados en terminal

Conclusiones

Modelo XGBoost

El modelo XGBoost exhibe un rendimiento excepcional en la clasificación de instancias tanto positivas como negativas, destacando por su capacidad notable para discriminar entre clases.

Ajuste de Submuestreo

Durante la investigación, se exploró el impacto del ajuste de submuestreo en el rendimiento del modelo XGBoost, lo que demostró ser una solución efectiva para abordar el desequilibrio de clases en el conjunto de datos.

Cambios en el modelo

Se sugiere la exploración de otras técnicas de modelado o ajustes de hiperparámetros con el fin de mejorar aún más la precisión y el recall.

Recomendaciones

- Utilizar los insights obtenidos del modelo predictivo para optimizar la inversión en publicidad digital.
- Implementar mejoras en el diseño y la navegación del sitio web para facilitar el proceso de compra y aumentar la retención de clientes.
- Continuar investigando y probando diferentes técnicas de modelado avanzado.
- Mantener el modelo predictivo actualizado con nuevos datos y cambios en el comportamiento del cliente.
- Utilizar las técnicas de segmentación para dirigir de manera más efectiva las campañas de marketing digital.
- Realizar un seguimiento regular del desempeño de las estrategias implementadas.
- Promover el uso de datos y análisis dentro de la empresa.
- Estar atento a las tendencias emergentes en el mercado de e-commerce.
- Promover la colaboración entre los diferentes departamentos.
- Establecer métricas claras para evaluar el desempeño de la estrategia de e-commerce.