
Realizado por:
Valentina Murgó

→
Predicción ventas de
CALLCENTER

Comisión:
61680

Abstracto con Motivación y Audiencia



Este proyecto tiene como objetivo analizar y visualizar el desempeño de las ventas realizadas por un Call Center durante el mes de enero de 2025. A través de la exploración de datos, buscamos identificar tendencias en las ventas, los productos más solicitados y la tasa de activaciones.

Se utilizarán técnicas de limpieza de datos, análisis exploratorio y visualización para extraer información valiosa que pueda ayudar en la toma de decisiones estratégicas.

Abstracto con Motivación y Audiencia



Este análisis está dirigido a diferentes actores clave dentro de la empresa, incluyendo:

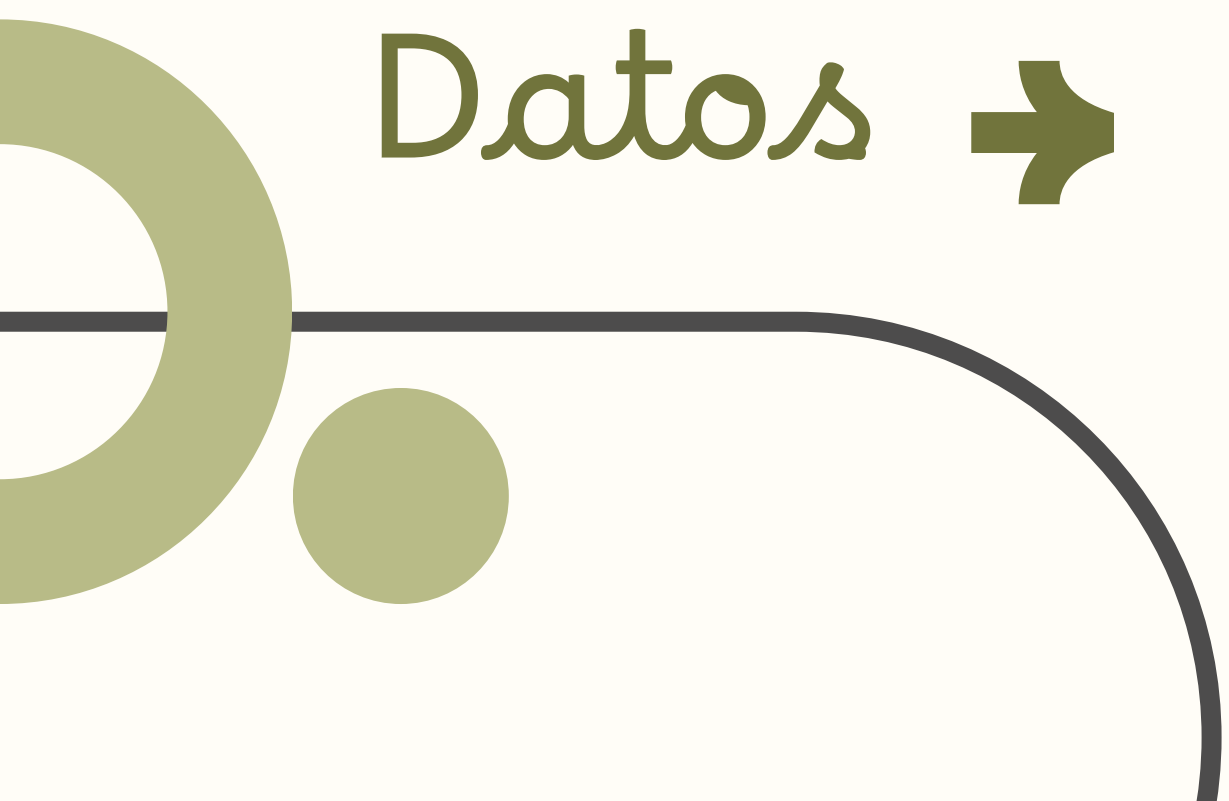
- Gerencia Comercial → Para identificar oportunidades de crecimiento y optimizar estrategias de venta.
 - Supervisores del Call Center → Para evaluar el rendimiento del equipo y mejorar los procesos de atención.
 - Analistas de Datos → Para generar reportes estratégicos.
-

Contexto Comercial y Analítico



El sector de telecomunicaciones es altamente competitivo, y las estrategias de ventas a través de call centers juegan un papel fundamental en la adquisición de clientes, fidelización y rentabilidad de las empresas.

Preguntas/ Hipótesis a Resolver mediante el Análisis de Datos ➔



Preguntas:

- ¿Cuál es la tasa de activación de las ventas realizadas?
- ¿Cuáles fueron los productos más vendidos?

Hipótesis:

- El producto más vendido es el más económico
 - La Tasa de activación es más ágil dependiendo del producto
-

→ Lectura de datos ←

Librerías necesarias para el análisis de datos

```
[ ] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
!pip install xgboost
import xgboost as xgb
from sklearn.model_selection import GridSearchCV
```

Configuración del entorno de Python en Google Colab

```
[ ] from google.colab import drive
import os

drive.mount("/content/drive")
print(os.getcwd())

os.chdir("/content/drive/My Drive/")
print(os.getcwd())
```

⇄ Mounted at /content/drive
/content
/content/drive/My Drive

Lectura del dataset

```
[ ] df = pd.read_csv('/content/drive/MyDrive/DataFrameVentas.csv', sep=";", encoding="latin1")
```

➔ Análisis inicial del Dataset ➔

```
[ ] df.shape
```

```
➔ (2035, 91)
```

Indica que mi dataset contiene 91 columnas y 2035 filas

```
[ ] df.head()
```



	Id	Fecha de Venta	Hora de carga	Fecha de Activación	Línea a Portar	Bandeja IRIS	Fecha de Portación	Fecha de aprobación	Usuario de la Línea	Producto Anterior	...	Producto actual correcto	Compañía actual correcta	Documento correcto	Domicilio correcta	Producto correcto	Información correcta
0	1433	15/01/2025	20/06/2023 15:22	16/01/2025	3.425767e+09	NaN	17/01/2025	16/01/2025	titular	POSPAGO	...	No	No	No	No	No	NaN
1	1577	14/01/2025	21/06/2023 20:02	14/01/2025	3.875072e+09	OSAR	NaN	NaN	""	POSPAGO	...	No	No	No	No	No	NaN
2	1704	21/01/2025	23/06/2023 13:18	22/01/2025	3.416390e+09	OSAR	24/01/2025	22/01/2025	""	POSPAGO	...	No	No	No	No	No	NaN
3	2895	22/01/2025	25/07/2023 11:30	NaN	1.122257e+09	NaN	NaN	NaN	1122257305	POSPAGO	...	No	No	No	No	No	NaN
4	3546	23/01/2025	07/08/2023 14:53	23/01/2025	1.149463e+09	NaN	27/01/2025	24/01/2025	""	POSPAGO	...	No	No	No	No	No	NaN

5 rows × 91 columns

➔ Análisis inicial del Dataset ➔

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2035 entries, 0 to 2034
Data columns (total 91 columns):
#   Column                               Non-Null Count  Dtype
---  -
0   Id                                   2035 non-null   int64
1   Fecha de Venta                      2035 non-null   object
2   Hora de carga                      2035 non-null   object
3   Fecha de Activación                1522 non-null   object
4   Línea a Portar                     2020 non-null   float64
5   Bandeja IRIS                      1669 non-null   object
6   Fecha de Portación                 1283 non-null   object
7   Fecha de aprobación                1288 non-null   object
8   Usuario de la Línea                2020 non-null   object
9   Producto Anterior                  2035 non-null   object
10  Plan actual                         0 non-null      float64
11  Producto                           2035 non-null   object
12  Precio                             2035 non-null   object
13  Cantidad de portas                 2035 non-null   int64
14  Estado de Solicitud                2035 non-null   object
15  Subestado                          174 non-null    object
16  Motivo de Rechazo                  147 non-null    object
17  Fecha de rechazo                   77 non-null     object
18  Fecha de reclamo                   86 non-null     object
19  Motivo de estado                   1825 non-null   object
```

```
20 PO                                1528 non-null   object
21 Nombre y Apellido del titular     2035 non-null   object
22 Apellido                           2035 non-null   object
23 DNI                                2035 non-null   int64
24 Vendedor                           2035 non-null   object
25 ID secundario                      2015 non-null   object
26 Ejecutivo                          2035 non-null   object
27 Supervisor                         2011 non-null   object
28 Envío                              2020 non-null   object
29 Número de Guía                     21 non-null     object
30 Número de Factura                  0 non-null      float64
31 Fuente de Solicitud                2035 non-null   object
32 Número de Orden                    1876 non-null   object
33 Observaciones                      2035 non-null   object
34 Provincia                          2035 non-null   object
35 Ciudad                             2035 non-null   object
36 Código Postal                     2035 non-null   object
37 Nexo                               2035 non-null   object
38 Fecha de Nacimiento                2022 non-null   object
39 Dirección                          2035 non-null   object
40 Número                             2035 non-null   object
41 Torre / Monoblock                  2035 non-null   object
42 Piso                              2035 non-null   object
43 Departamento                       2035 non-null   object
44 Entre Calles                       2035 non-null   object
45 Barrio                             158 non-null    object
46 Manzana                           2035 non-null   object
47 Casa / Lote                       2035 non-null   object
48 Otras referencias                  15 non-null     object
49 Referencias                        2035 non-null   object
50 Teléfono Principal (Fijo)           2035 non-null   object
51 Teléfono Adicional (Celular)        2035 non-null   object
52 Teléfono Adicional #2 (Celular)     2035 non-null   object
53 Número de Chip                     2020 non-null   object
54 Código de Área                     1047 non-null   float64
55 Prefijo                           1047 non-null   float64
56 Compañía de Teléfono                2020 non-null   object
57 Fecha de cita                       12 non-null     object
```

```
58 Hora de cita                       15 non-null     object
59 Comentarios                         2035 non-null   object
60 Número de PIN                       1566 non-null   float64
61 CHIP en mano                       2035 non-null   object
62 Totalización cargada en web         2035 non-null   object
63 Fecha de envío de Chip               2 non-null      object
64 Fecha entrega SIM                   1 non-null      object
65 Tipo de confirmación SIM            2020 non-null   object
66 Fecha llamado confirmación SIM       0 non-null      float64
67 Fecha de Vencimiento de Pin         1567 non-null   object
68 Usuario logística                   495 non-null    object
69 Comentarios de logística            2020 non-null   object
70 Días transcurridos desde             2035 non-null   int64
71 Fecha de cambio de estado           2035 non-null   object
72 ANI fijo asignado                   15 non-null     object
73 centralizador_ftth_latitud           2013 non-null   float64
74 centralizador_ftth_longitud          2013 non-null   float64
75 centralizador_ftth_ctos_cercanos     2013 non-null   float64
76 centralizador_fecha_sincronizacion   2015 non-null   object
77 Formulario de calidad completado     1 non-null      object
78 Formulario de calidad exitoso        2035 non-null   object
79 Línea a portar correcta              2035 non-null   object
80 Titular de línea correcta            2035 non-null   object
81 Producto actual correcto             2035 non-null   object
82 Compañía actual correcta             2035 non-null   object
83 Documento correcto                  2035 non-null   object
84 Domicilio correcta                  2035 non-null   object
85 Producto correcto                   2035 non-null   object
86 Informacion correcta                 0 non-null      float64
87 Email calidad                       2 non-null      object
88 Interesado en fibra                 2035 non-null   object
89 Dirección de fibra                   0 non-null      float64
90 Link Google Maps                    2035 non-null   object
dtypes: float64(12), int64(4), object(75)
memory usage: 1.4+ MB
```

➔ Análisis inicial del Dataset ➔

Descripción de las variables

Id: Número de id de la carga de venta.

Fecha de venta: fecha en la que se carga la venta

Hora de carga: hora en la que se carga la venta

Fecha de Activación: fecha en la que se activa la portabilidad

Linea a portar: número de teléfono del cliente

Bandeja IRIS: sistema donde fue cargado

Fecha de portación: fecha de portabilidad asignada

Fecha de aprobación: fecha de aprobacion asignada

Usuario de la linea: número del titular de la línea

Producto anterior: tipo de producto anterior

Plan actual:

Producto: producto que adquiere en esta compañía

Precio:

Cantidad de portas: cantidad de números del cliente que realizan la portabilidad

Estado de Solicitud: estado administrativo en el que se encuentra el trámite

Subestado: para alguna aclaración necesaria

Motivo de Rechazo: si la solicitud fuese rechazada, se agregan los motivos

Fecha de rechazo: se agrega fecha de dicho rechazo

PO: Número de PO que asigna Movistar

Nombre y Apellido del titular: Aparece solo el nombre del titular

Apellido: indica el apellido del titular

DNI: número de identificación del titular

Vendedor: agente que realizó la venta

ID secundario: ID del agente

Ejecutivo: entidad que realizó la venta

Supervisor: lider del agente

Envio: medio por el que se realiza el envio

Número de Guia: número de seguimiento

Número de factura:

Fuente de Solicitud: fuente donde recibe el dato el vendedor

Número de Orden: aclaración entre los administrativos

Observaciones: celda para agregar aclaraciones

Provincia: Provincia del envio

Ciudad: Ciudad del envio

Código Postal: CP del envio

Fecha de Nacimiento: fecha de nacimiento del titular

Dirección: dirección del envio

Número: número de la dirección del envio

Piso: Piso del envio

Departamento: Departamento del envio

Entre Calles: Entre calles del envio

Barrio: Barrio del envio

Manzana: Manzana del envio

Casa/Lote: Casa del envio

Otras referencias: Referencias adicionales

Referencias: Referencias adicionales

Teléfono Principal: Teléfono principal del titular

Teléfono Adicional: Teléfono Adicional del titular

Teléfono Adicional 2: Teléfono Adicional conviviente del titular

Numero de Chip: Número asignado de CHIP

Código de area: Código de área del titular

Prefijo: Prefijo del número del titular

Compañía de Teléfono: compañía de teléfono anterior

Fecha de Cita: fecha retiro en sucursal

Hora de Cita: hora retiro en sucursal

Comentarios: comentarios adicionales

Número de PIN: código de portabilidad asignado

CHIP en mano: indica si el chip fue recibido

→ Data Wrangling - Limpieza y transformación de datos ←

```
[ ] df.isnull().sum()
```

	0
Id	0
Fecha de Venta	0
Hora de carga	0
Fecha de Activación	513
Linea a Portar	15
...	...
Informacion correcta	2035
Email calidad	2033
Interesado en fibra	0
Dirección de fibra	2035
Link Google Maps	0

91 rows x 1 columns

Identifico que no tengo ningún valor duplicado

```
[ ] df.duplicated().sum()
```

```
np.int64(0)
```

← Reviso la cantidad de valores nulos y en que columnas se encuentran.

➔ Data Wrangling - Limpieza y transformación de datos ➔

```
columnas_a_eliminar = [  
    "Id",  
    "Formulario de calidad completado", "Línea a portar correcta",  
    "Titular de línea correcto", "Producto actual correcto",  
    "Compañía actual correcta", "Documento correcto",  
    "Domicilio correcta", "Producto correcto",  
    "Email calidad", "Interesado en fibra",  
    "Dirección de fibra", "Link Google Maps",  
    "Fecha de envío de Chip", "Fecha entrega SIM",  
    "Tipo de confirmación SIM", "Fecha de vencimiento de Pin",  
    "Usuario logística", "Comentarios de logística",  
    "Fecha de cambio de estado", "ANI fijo asignado",  
    "Fecha de cita", "Hora de cita",  
    "Teléfono Principal (Fijo)",  
    "Teléfono Adicional (Celular)",  
    "Teléfono Adicional #2 (Celular)",  
    "Dirección", "Número",  
    "Torre / Monoblock", "Piso",  
    "Departamento", "Entre Calles",  
    "Barrio", "Manzana",  
    "Casa / Lote", "Otras referencias",  
    "Referencias", "Nexo",  
    "Fecha de Nacimiento", "Envío",  
    "Número de Guía", "Número de Orden",  
    "Observaciones", "Comentarios",  
    "Precio"  
]  
  
df_clear = df.drop(columns=columnas_a_eliminar, errors='ignore')
```



```
[ ] df_clear = df.dropna(axis=1, how='all')
```

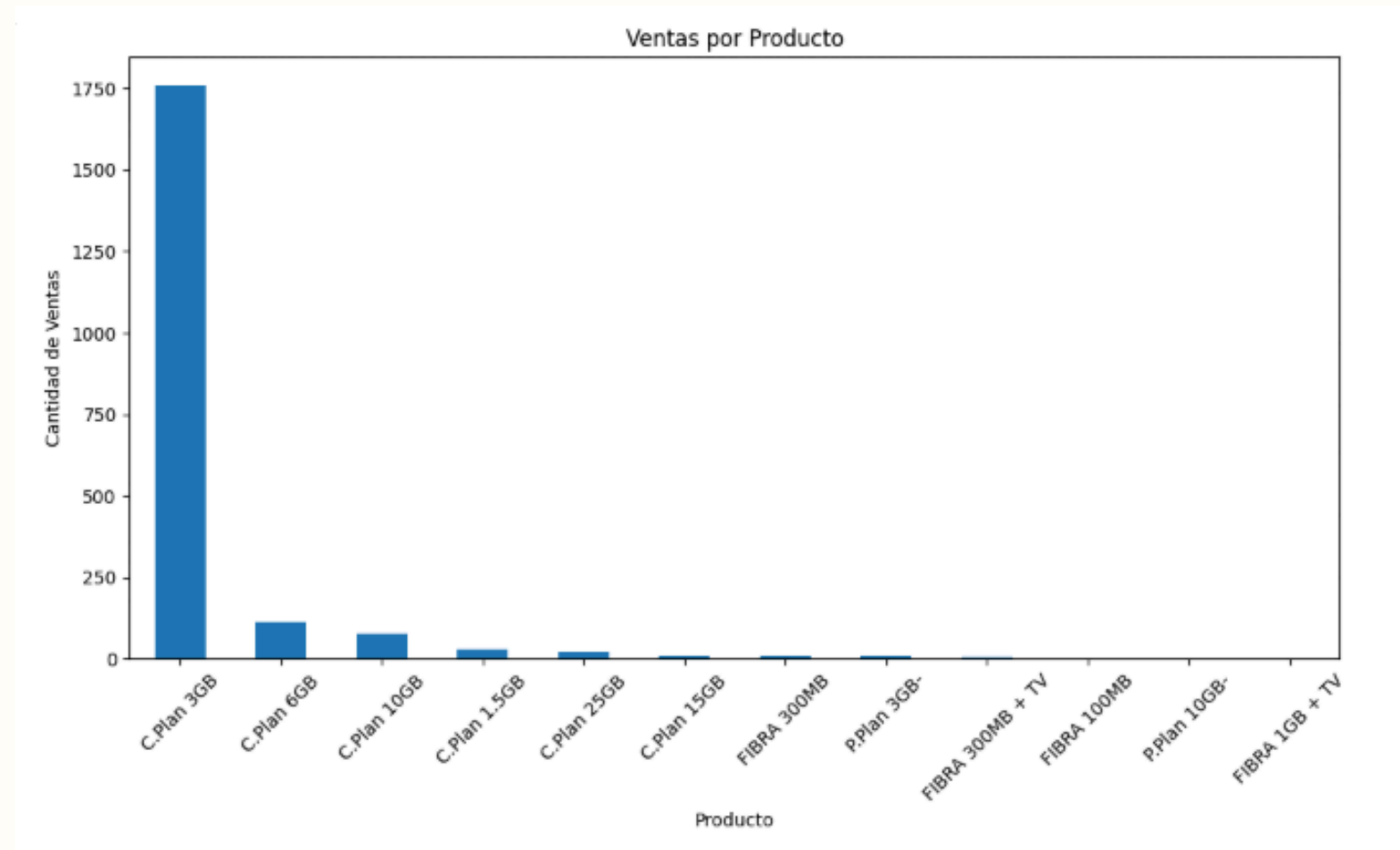
Después de revisar los valores, decido eliminar únicamente las columnas que presentan la totalidad de sus valores nulos.



Elimino todas las columnas que interpreto que no son útiles para analizar y que entorpecen el análisis de mi dataset.

Tratamiento de valores outliers

```
[ ] df_clear["Producto"].value_counts().plot(kind="bar", figsize=(12,6), title="Ventas por Producto")
plt.xlabel("Producto")
plt.ylabel("Cantidad de Ventas")
plt.xticks(rotation=45)
plt.show()
```



El valor del outlier se encuentra en el plan de 3GB, pero decido no eliminarlo o ajustarlo con la moda, ya que representa un caso real y significativo: ventas de productos altamente demandados.

Tratamiento de valores outliers

Reviso su estado administrativo para saber si realmente son ventas aprobadas y confirmo mi decisión de mantener este outlier

```
df_clear[df_clear["Producto"].str.contains("3GB", na=False)]["Estado de Solicitud"].value_counts()
```

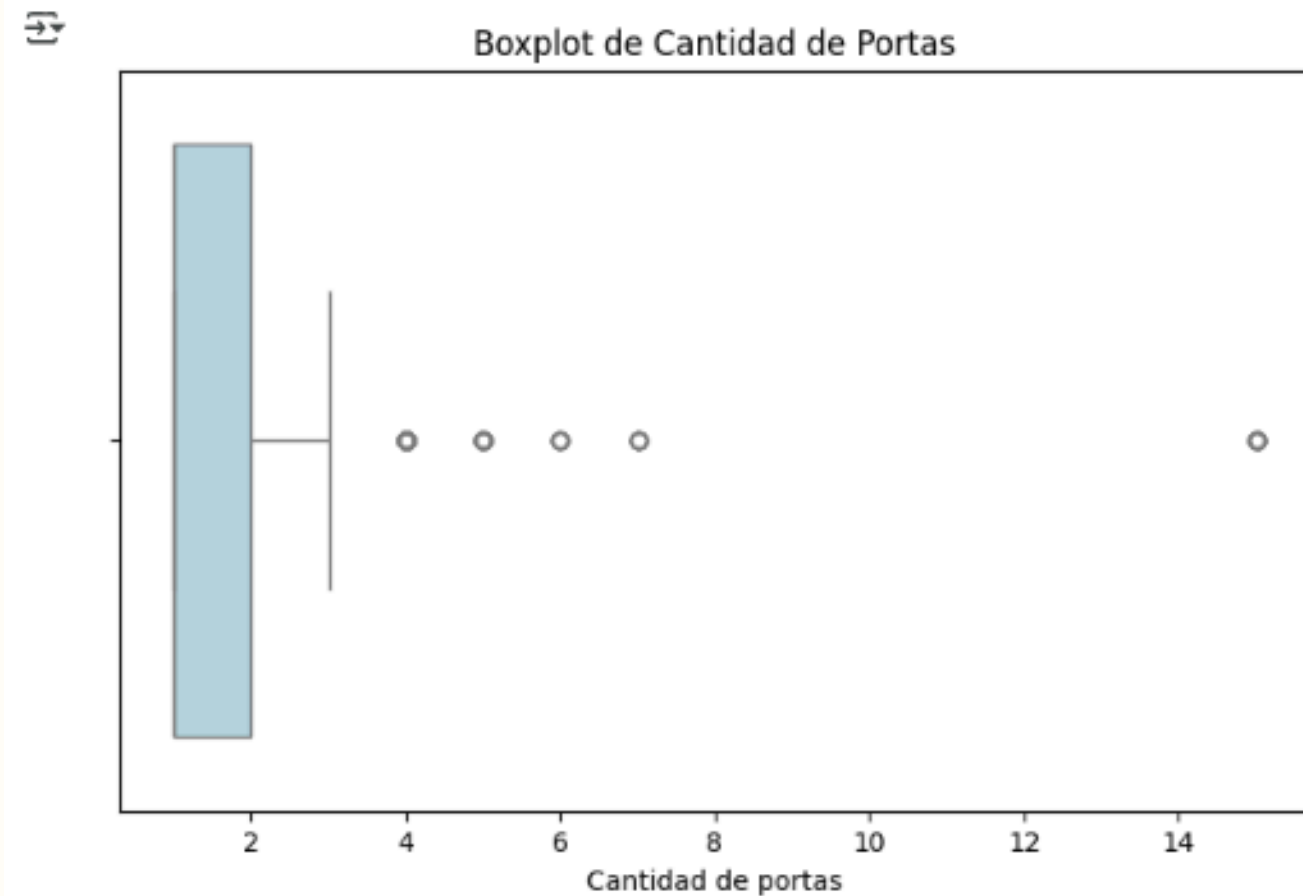
Estado de Solicitud	count
APROBADA	1113
NO CUMPLE PERMANENCIA	156
CANCELADA	151
PENDIENTE DE OBSERVACION	105
RETENCION EMPRESA ACTUAL	70
LOGISTICA	40
ENVIADA A AG	32
PEDIR PIN - SIM ENTREGADA	25
SIM EN CUSTODIA	18
ACTIVADA	11
EXCEDE TIEMPO EN BANDEJA	8
GENERADA POR OTRO AGENTE	7
CORPO CANCELADO	7
GESTION DE RECHAZO	5
CORPO APROBADO	4
RECLAMADA	3
CORPO POSTVENTA	3
AGUARDANDO PAGO DEUDA	2
CORPO EN TRAMITE (TELEFONICA)	2
RECHAZADA	2
CORPO RECLAMADO	1
POSTVENTA	1

dtype: int64

➔ Análisis Exploratorio de datos ➔

Análisis univariado

```
[ ]  
plt.figure(figsize=(8,5))  
sns.boxplot(x=df["Cantidad de portas"], color="lightblue")  
plt.title("Boxplot de Cantidad de Portas")  
plt.show()
```



La mayor parte de los datos se encuentran entre 1 y 3 portas vendidas y su mediana es de 2 porta. El rango intercuartílico indica que la mayoría se encuentran en un rango bajo. Se observan varios outliers mayores a 3, pero uno que destaca de 15.

➔ Análisis Exploratorio de datos ➔

Análisis univariado

```
df_clear[df_clear["Cantidad de portas"] == 15]["Estado de Solicitud"]
```

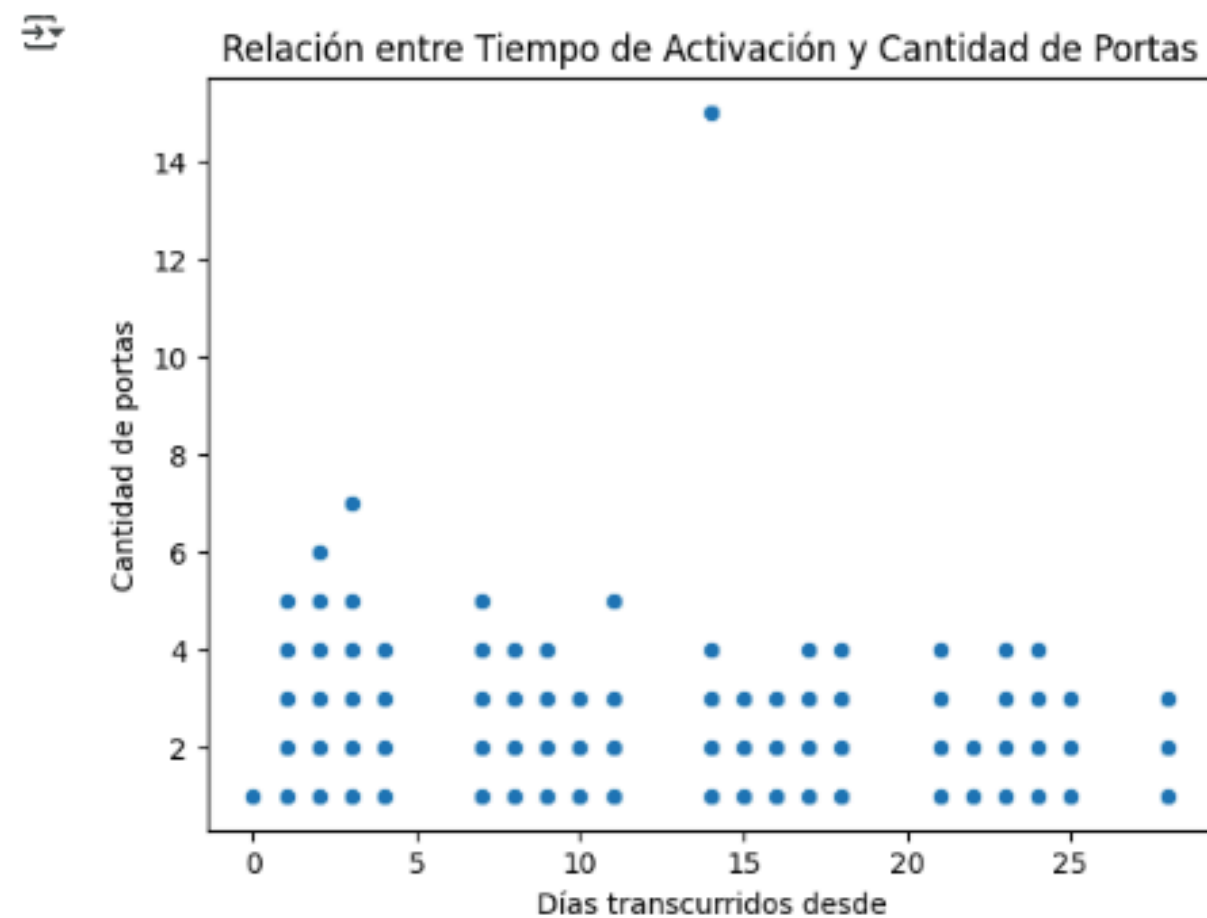
	Estado de Solicitud
1095	CORPO EN TRAMITE (TELEFONICA)
1096	CORPO EN TRAMITE (TELEFONICA)
1097	CORPO EN TRAMITE (TELEFONICA)
1099	CORPO EN TRAMITE (TELEFONICA)
1102	CORPO EN TRAMITE (TELEFONICA)
1103	CORPO EN TRAMITE (TELEFONICA)
1104	CORPO EN TRAMITE (TELEFONICA)
1106	CORPO EN TRAMITE (TELEFONICA)
1107	CORPO EN TRAMITE (TELEFONICA)
1108	CORPO EN TRAMITE (TELEFONICA)
1109	CORPO EN TRAMITE (TELEFONICA)
1110	CORPO EN TRAMITE (TELEFONICA)
1111	CORPO EN TRAMITE (TELEFONICA)
1119	CORPO EN TRAMITE (TELEFONICA)

El outlier no es un error, sino que corresponde a clientes corporativos que portan múltiples líneas en una sola transacción. Esto explica el valor atípico en la variable "Cantidad de portas".

➔ Análisis Exploratorio de datos ➔

Análisis bivariado

```
sns.scatterplot(x=df_clear["Días transcurridos desde"], y=df_clear["Cantidad de portas"])  
plt.title("Relación entre Tiempo de Activación y Cantidad de Portas")  
plt.show()
```



Relación entre Tiempo de Activación y Cantidad de Portas

La activación suele completarse en periodos de 0 a 10 días, con algunos casos extendiéndose hasta los 25 días.

Nuevamente aparecen las 15 portas como outlier, pero esta vez en tiempo de activación, lo que sugiere que los clientes corporativos tardan más en activar que la porta individuos. Para los demás casos, no hay un patrón claro que indique que más portas = mayor tiempo de activación.

➔ Análisis Exploratorio de datos ➔

Análisis multivariado

```
[ ]
pivot_table = df_clear.pivot_table(index="Vendedor", columns="Producto", values="Cantidad de portas", aggfunc="sum")

pivot_table_top10 = pivot_table.head(10)

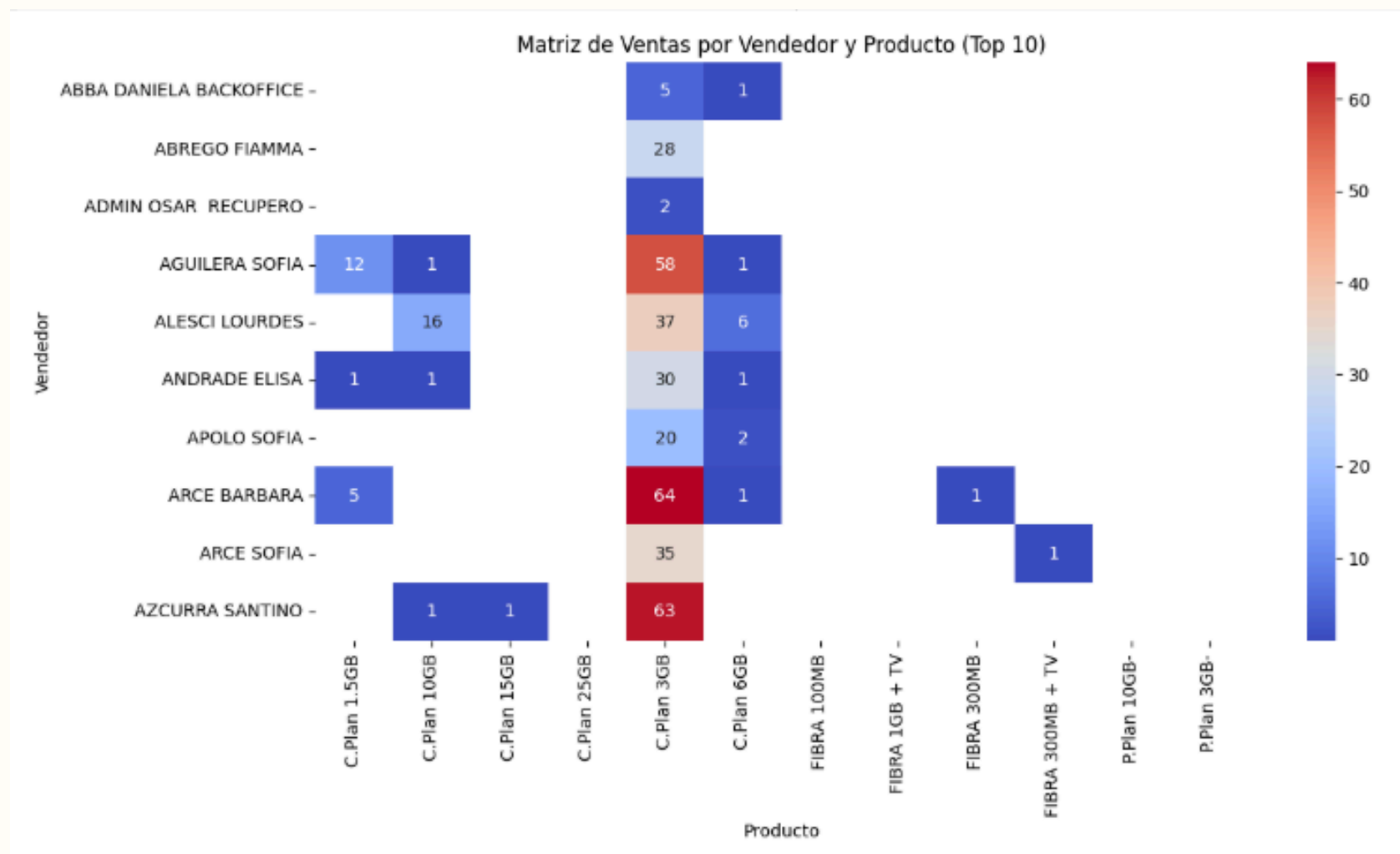
plt.figure(figsize=(12,6))
sns.heatmap(pivot_table_top10, cmap="coolwarm", annot=True, fmt=".0f")
plt.title("Matriz de Ventas por Vendedor y Producto (Top 10)")
plt.show()
```

Realicé una muestra de 10 ejemplos debido a la alta cantidad de vendedores.

-Se puede ver claramente que el plan más vendido es el de 3GB en todos los casos.

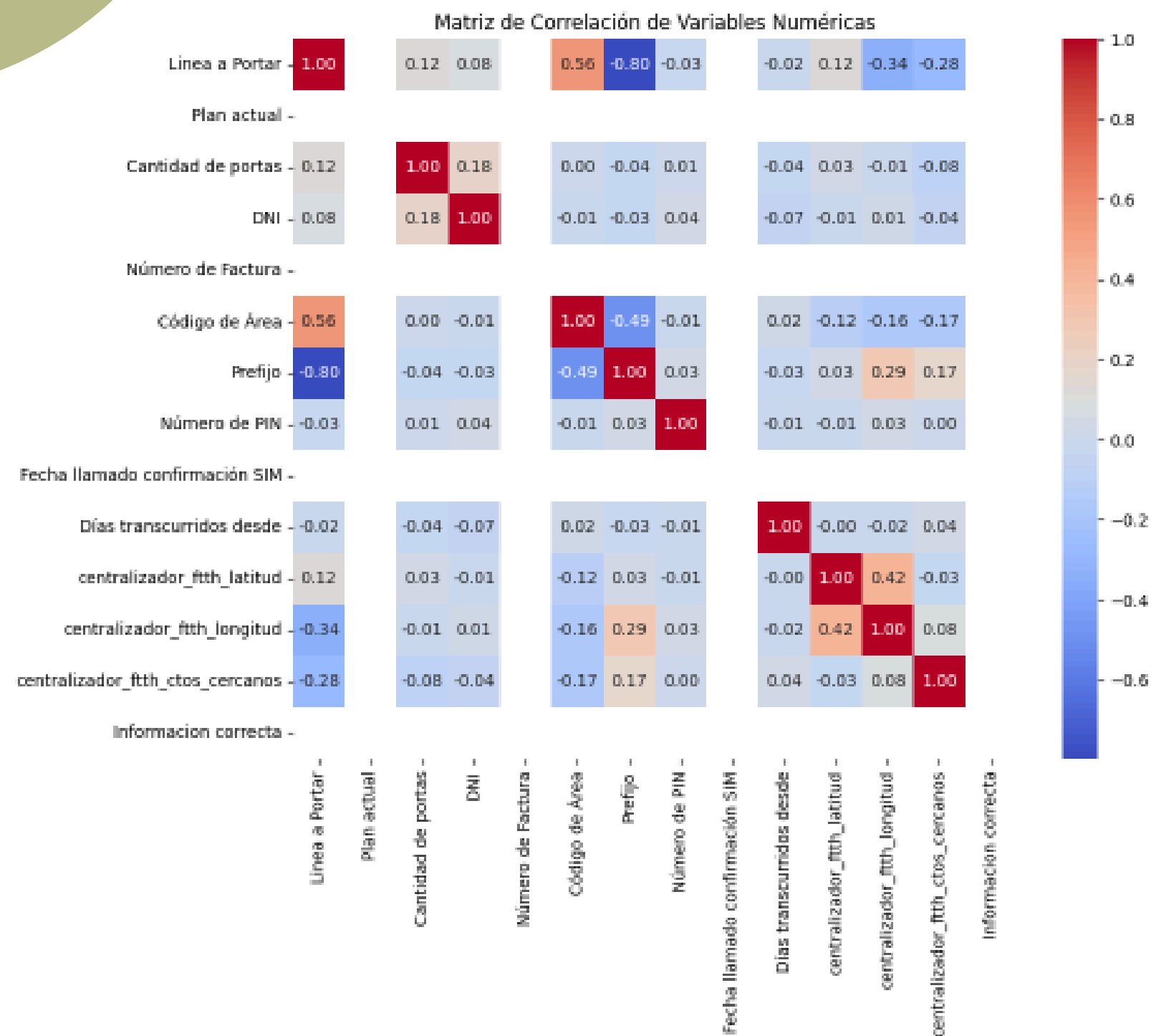
-La vendedora Lourdes Alesci destaca vendiendo la mayor cantidad de planes de 5GB

-Se vende una mínima cantidad de fibra y tv y además algunos vendedores no venden ninguno de estos productos.



➔ Análisis Exploratorio de datos ➔

Matriz de correlación



```
[ ] # Seleccionar solo las columnas numéricas
numeric_cols = df_clear.select_dtypes(include=['float64', 'int64']).columns

# Calcular la matriz de correlación
corr_matrix = df_clear[numeric_cols].corr()

# Visualizar la matriz de correlación con un mapa de calor
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f', cbar=True)
plt.title('Matriz de Correlación de Variables Numéricas')
plt.show()
```

"Código de Área" y "Prefijo" tienen correlación perfecta ya que el código de área y prefijo telefónico están directamente relacionados, por lo tanto, una de estas variables podría eliminarse del análisis así no es redundante.

Se puede analizar si hay un patrón en los "Días transcurridos desde" y "Confirmación SIM" para optimizar los tiempos de espera.



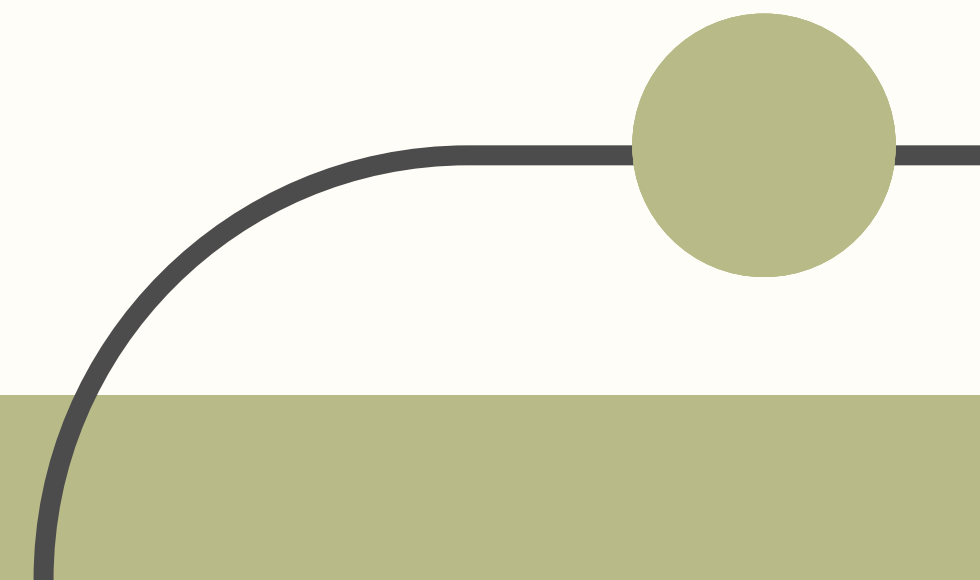
➔ Conclusiones del EDA ➔

Se observa que la mayoría de las ventas están concentradas en ciertos productos específicos (ej. Plan 3GB).

Hay una relación entre cantidad de ventas y tasa de activación, donde algunos vendedores convierten más ventas en activaciones exitosas.

Recomendación: Diversificar la oferta de productos → Incentivar ventas de planes menos vendidos (como fibra y TV).

Diferenciar estrategias para clientes corporativos → Su proceso de compra es diferente al de clientes individuales.



➔ Preprocesamiento de datos ➔

Codificación de variables categóricas

```
df_clear['Vendedor_encoded']
```



Vendedor_encoded

0	75
1	3
2	26
3	34
4	54
...	...
2030	35
2031	52
2032	52
2033	52
2034	52

2035 rows × 1 columns

dtype: int64

```
[ ] le = LabelEncoder()  
df_clear['Vendedor_encoded'] = le.fit_transform(df_clear['Vendedor'])
```

A través del LabelEncoding convierto la variable "Vendedor" en una variable numérica

➔ Preprocesamiento de datos ➔

Codificación de variables categóricas

```
[19] # Paso 1: Generar columnas OneHotEncoding a partir de 'Supervisor' (usando el df original)
      supervisor_ohe = pd.get_dummies(df['Supervisor'], prefix='Supervisor')

      # Paso 2: Agregar las columnas nuevas al dataframe limpio df_clear
      df_clear = pd.concat([df_clear, supervisor_ohe], axis=1)
```

Converti la variable "Supervisor" en una variable numérica con OneHotEncoding y agregue las columnas a mi dataframe

➔ Preprocesamiento de datos ➔

Estandarización

```
df_clear['Fecha de Venta'] = pd.to_datetime(df['Fecha de Venta'], errors='coerce', dayfirst=True)
df_clear['Fecha de Nacimiento'] = pd.to_datetime(df['Fecha de Nacimiento'], errors='coerce', dayfirst=True)
df_clear['Edad'] = df_clear.apply(
    lambda row: row['Fecha de Venta'].year - row['Fecha de Nacimiento'].year
    if pd.notnull(row['Fecha de Venta']) and pd.notnull(row['Fecha de Nacimiento']) else None, axis=1
)
df_clear['Edad'] = df_clear['Edad'].fillna(df_clear['Edad'].median())
```

Primero creo una nueva columna con la edad de los clientes

➔ Preprocesamiento de datos ➔

Estandarización

```
cols = ['Edad', 'Cantidad de portas', 'Días transcurridos desde']

# Crear versiones escaladas
original = df_clear[cols].copy()
standard_scaled = pd.DataFrame(StandardScaler().fit_transform(original), columns=cols)
robust_scaled = pd.DataFrame(RobustScaler().fit_transform(original), columns=cols)
```

Estandarizo utilizando 'Standard Scaled y RobustScaler para comparar

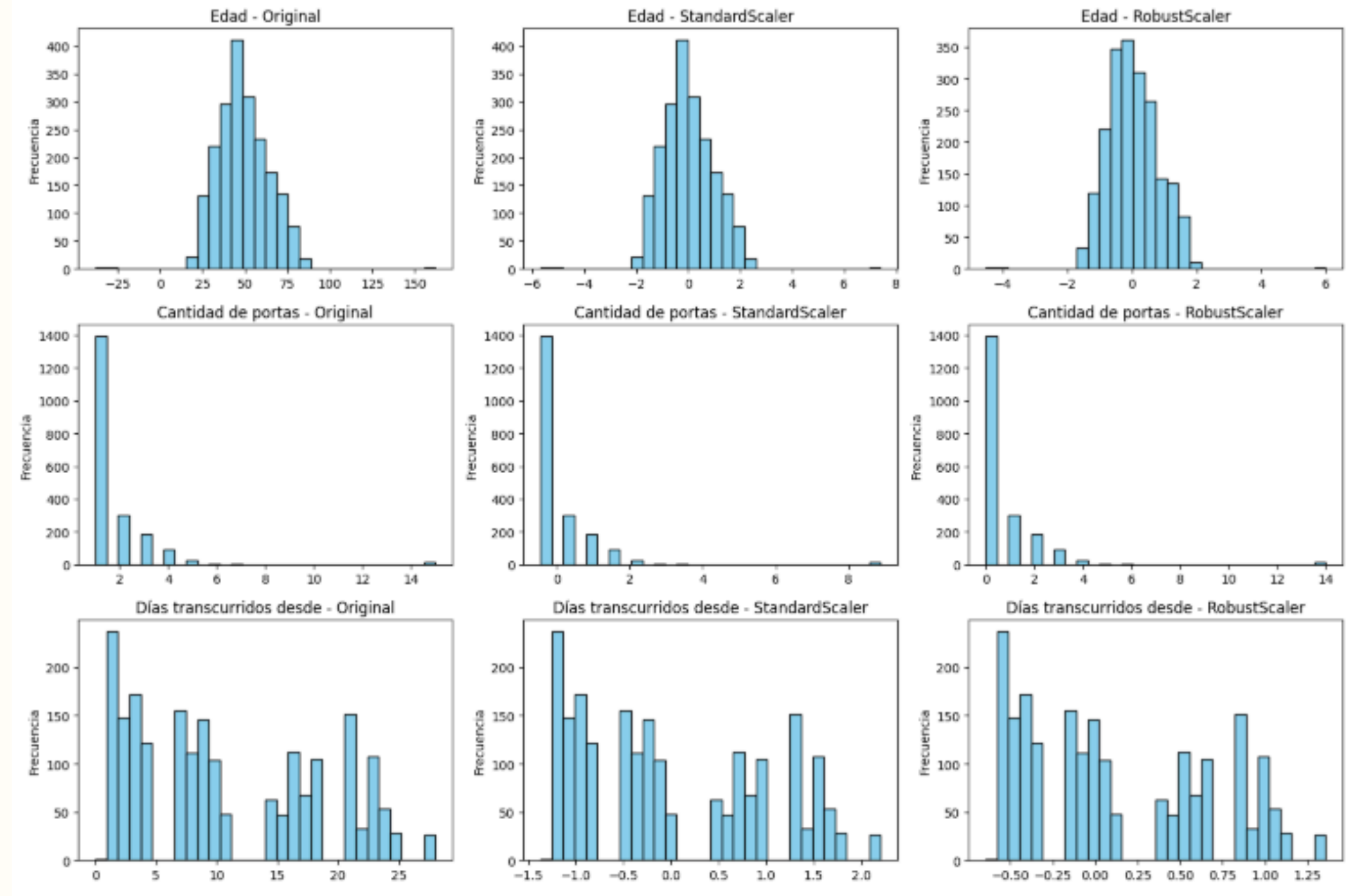
```
# Crear gráfico comparativo
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(15, 10))
titles = ['Original', 'StandardScaler', 'RobustScaler']
datasets = [original, standard_scaled, robust_scaled]

for i, col in enumerate(cols):
    for j, dataset in enumerate(datasets):
        axes[i, j].hist(dataset[col], bins=30, color='skyblue', edgecolor='black')
        axes[i, j].set_title(f'{col} - {titles[j]}')
        axes[i, j].set_ylabel('Frecuencia')

plt.tight_layout()
plt.show()
```


➔ Preprocesamiento de datos ➔

Estandarización



➔ Preprocesamiento de datos ➔

Estandarización

```
# Escalar con StandardScaler  
scaler_std = StandardScaler()  
df_clear['Edad'] = scaler_std.fit_transform(df_clear[['Edad']])
```

```
# Escalar con RobustScaler  
scaler_robust = RobustScaler()  
df_clear[['Cantidad de portas', 'Días transcurridos desde']] = scaler_robust.fit_transform(  
    df_clear[['Cantidad de portas', 'Días transcurridos desde']]  
)
```

Decido escalar 'Edad' con Standard Scaler ya que tiene una distribución normal, sin outliers extremos, y a 'Cantidad de portas' (contiene outliers reales) y 'Días transcurridos desde' (contiene distribución sesgada) con Robust Scaler para no distorsionar el modelo

➔ Feature Selection ➔

Selección de variable objetivo y variables independientes

```
[59] # Estandarizar texto en minúsculas y quitar espacios extra
      df['Estado de Solicitud'] = df['Estado de Solicitud'].str.strip().str.lower()

      # Crear columna binaria: 1 si es 'aprobada', 0 en cualquier otro caso
      df_clear['Estado_binario'] = df['Estado de Solicitud'].apply(lambda x: 1 if x == 'aprobada' else 0)
```

Primero transformo mi variable ‘Estado de Solicitud’ en una variable binaria llamada ‘Estado binario’ para poder analizar su correlación

➔ Feature Selection ➔

Selección de variable objetivo y variables independientes

```
# Seleccionar solo columnas numéricas
numericas = df_clear.select_dtypes(include=['int64', 'float64'])

# Calcular matriz de correlación
correlacion = numericas.corr()

# Graficar heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlacion, annot=True, fmt=".2f", cmap='coolwarm', linewidths=0.5)
plt.title("📊 Matriz de Correlación de Variables Numéricas")
plt.tight_layout()
plt.show()
```

Creo la matriz de correlación para detectar relaciones importantes entre mis variables, es decir, las variables independientes que voy a utilizar en mi modelo



- **'Días transcurridos desde':** A mayor cantidad de días, más probabilidad de aprobación (¿posiblemente por ventas corporativas o validadas?).
- **'Edad':** Ligera correlación: edad más alta, mayor aprobación.
- **'Cantidad de portas':** A más portas, mayor chance de aprobación (quizá ventas grupales con validación más fuerte).



Modelos



Librerías necesarias para
implementar los modelos

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report
!pip install xgboost
import xgboost as xgb
```



Modelos



Random Forest Classifier

División de datos en conjuntos de entrenamiento y prueba

```
variables_utiles = [  
    'Edad',  
    'Cantidad de portas',  
    'Días transcurridos desde',  
]  
  
# Variables predictoras y objetivo  
X = df_clear[variables_utiles]  
y = df_clear['Estado_binario']  
  
# División de datos  
X_train, X_test, y_train, y_test = train_test_split(  
    X, y, test_size=0.2, stratify=y, random_state=42  
)
```



Modelos



Random Forest Classifier

División de datos en conjuntos de entrenamiento y prueba

```
[69] # Modelo Random Forest  
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)  
rf_model.fit(X_train, y_train)
```



RandomForestClassifier ⓘ ?
RandomForestClassifier(random_state=42)

Predicción con conjunto de prueba

```
[70]  
y_pred = rf_model.predict(X_test)
```




Modelos



Random Forest Classifier

Evaluación del rendimiento del modelo

```
print("📊 Matriz de Confusión:")
print(confusion_matrix(y_test, y_pred))

print("\n📋 Reporte de Clasificación:")
print(classification_report(y_test, y_pred))
```

📊 Matriz de Confusión:

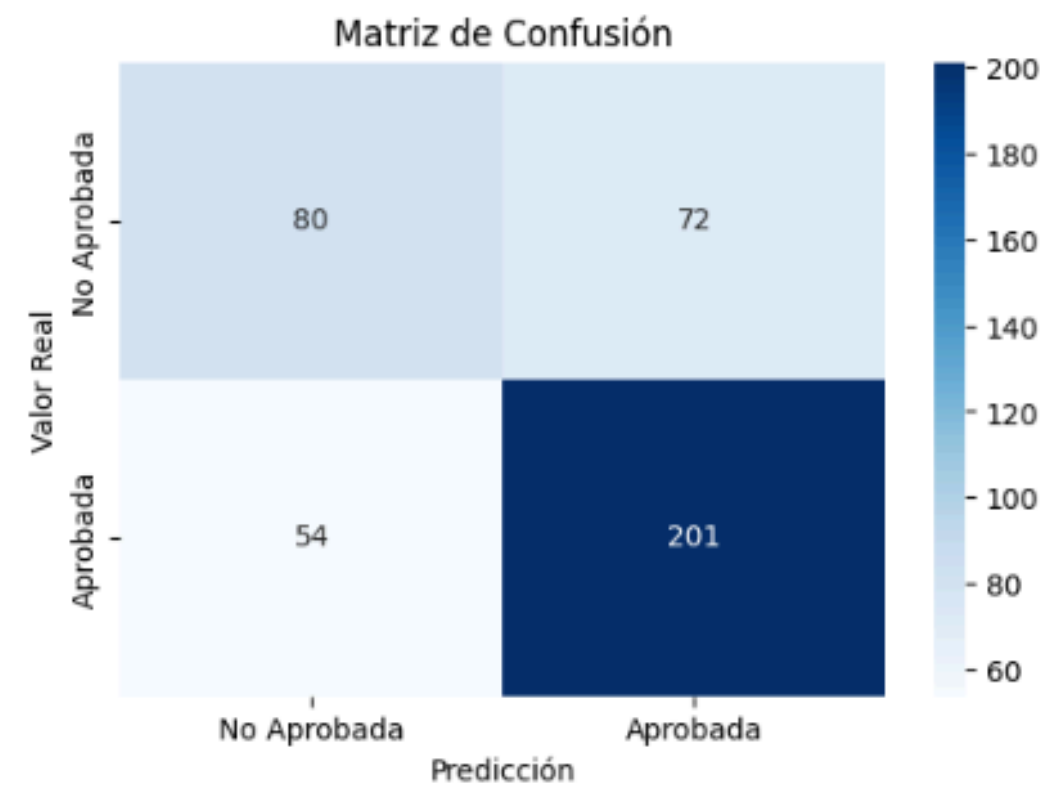
```
[[ 80  72]
 [ 54 201]]
```

📋 Reporte de Clasificación:

	precision	recall	f1-score	support
0	0.60	0.53	0.56	152
1	0.74	0.79	0.76	255
accuracy			0.69	407
macro avg	0.67	0.66	0.66	407
weighted avg	0.68	0.69	0.69	407

```
cm = confusion_matrix(y_test, y_pred)
labels = ['No Aprobada', 'Aprobada']

plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels, yticklabels=labels)
plt.title('Matriz de Confusión')
plt.xlabel('Predicción')
plt.ylabel('Valor Real')
plt.show()
```





Modelos



XGBoost

División de datos en conjuntos de entrenamiento y prueba

1. Cargar el modelo

```
xgb_model = xgb.XGBClassifier(  
    use_label_encoder=False,  
    eval_metric='logloss',  
    random_state=42  
)
```

2. Entrenar el modelo con tus datos

```
xgb_model.fit(X_train, y_train)
```

División de datos

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



Modelos



XGBoost

Predicción con conjunto de prueba

```
y_pred_xgb = xgb_model.predict(x_test)
```

Evaluación del rendimiento del modelo

```
print("📊 Matriz de Confusión - XGBoost:")  
print(confusion_matrix(y_test, y_pred_xgb))  
  
print("\n📋 Reporte de Clasificación - XGBoost:")  
print(classification_report(y_test, y_pred_xgb))
```



Modelos



XGBoost

Evaluación del rendimiento del modelo

 Matriz de Confusión - XGBoost:

```
[[ 77  75]
 [ 61 194]]
```

 Reporte de Clasificación - XGBoost:

	precision	recall	f1-score	support
0	0.56	0.51	0.53	152
1	0.72	0.76	0.74	255
accuracy			0.67	407
macro avg	0.64	0.63	0.64	407
weighted avg	0.66	0.67	0.66	407



Modelos

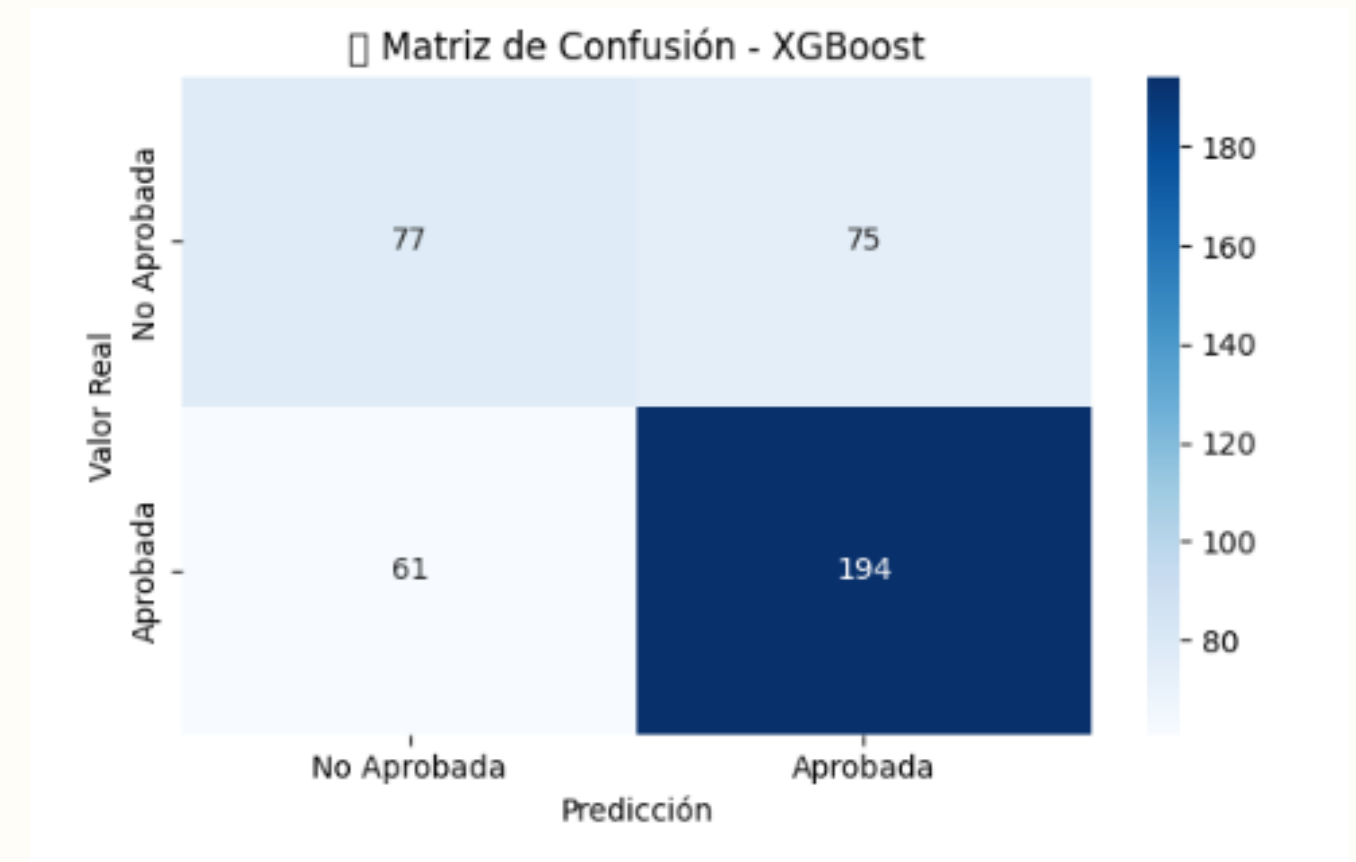


XGBoost

Evaluación del rendimiento del modelo

```
cm = confusion_matrix(y_test, y_pred_xgb)
labels = ['No Aprobada', 'Aprobada']

# Graficar la matriz
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels, yticklabels=labels)
plt.title("Matriz de Confusión - XGBoost")
plt.xlabel("Predicción")
plt.ylabel("Valor Real")
plt.tight_layout()
plt.show()
```





Modelos



Conclusión sobre el modelado y las metricas

- **Random Forest**
Precisión general (mejor accuracy, precision, recall, y f1 en ambas clases)
Predice mejor ambas clases sin perder mucho balance
Mejora ligeramente la recuperación de aprobadas (Clase 1)
- **XGBoost**
Tiene un poquito más de falsos negativos y falsos positivos
Es más sensible al ajuste de hiperparámetros (lo podemos mejorar)
Puede rendir mejor con más datos o más features

➔ Optimización de modelos ➔

Aplico hiperparámetros con GridSearchCV sobre el modelo de XGBoost

```
from sklearn.model_selection import GridSearchCV

# Crear el modelo base
xgb_clf = xgb.XGBClassifier(
    use_label_encoder=False,
    eval_metric='logloss',
    random_state=42
)

# Definir el espacio de búsqueda
param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [3, 5],
    'learning_rate': [0.01, 0.1],
    'subsample': [0.8],
    'colsample_bytree': [0.8]
}

# Configurar GridSearchCV
grid_search = GridSearchCV(
    estimator=xgb_clf,
    param_grid=param_grid,
    scoring='f1',
    cv=3,
    verbose=1,
    n_jobs=-1
)
```

```
# Ejecutar búsqueda
grid_search.fit(X_train, y_train)

# Ver mejores parámetros
print("🔧 Mejores hiperparámetros encontrados:")
print(grid_search.best_params_)

# Entrenar modelo final con esos parámetros
best_xgb = grid_search.best_estimator_
y_pred_best = best_xgb.predict(X_test)

# Evaluación
from sklearn.metrics import confusion_matrix, classification_report
print("\n📊 Matriz de Confusión:")
print(confusion_matrix(y_test, y_pred_best))
print("\n📋 Reporte de Clasificación:")
print(classification_report(y_test, y_pred_best))
```

➔ Optimización de modelos ➔

Aplico hiperparámetros con GridSearchCV sobre el modelo de XGBoost

```
from sklearn.model_selection import GridSearchCV
# Crear el modelo base
xgb_clf = xgb.XGBClassifier(
    use_label_encoder=False,
    eval_metric='logloss',
    random_state=42
)

# Definir el espacio de búsqueda
param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [3, 5],
    'learning_rate': [0.01, 0.1],
    'subsample': [0.8],
    'colsample_bytree': [0.8]
}

# Configurar GridSearchCV
grid_search = GridSearchCV(
    estimator=xgb_clf,
    param_grid=param_grid,
    scoring='f1',
    cv=3,
    verbose=1,
    n_jobs=-1
)
```

```
# Ejecutar búsqueda
grid_search.fit(X_train, y_train)

# Ver mejores parámetros
print("🔧 Mejores hiperparámetros encontrados:")
print(grid_search.best_params_)

# Entrenar modelo final con esos parámetros
best_xgb = grid_search.best_estimator_
y_pred_best = best_xgb.predict(X_test)

# Evaluación
from sklearn.metrics import confusion_matrix, classification_report
print("\n📊 Matriz de Confusión:")
print(confusion_matrix(y_test, y_pred_best))
print("\n📋 Reporte de Clasificación:")
print(classification_report(y_test, y_pred_best))
```

📊 Matriz de Confusión:

```
[[ 36 116]
 [ 10 245]]
```

📋 Reporte de Clasificación:

	precision	recall	f1-score	support
0	0.78	0.24	0.36	152
1	0.68	0.96	0.80	255
accuracy			0.69	407
macro avg	0.73	0.60	0.58	407
weighted avg	0.72	0.69	0.63	407

➔ Optimización de modelos ➔

Conclusión sobre la optimización

- Recupera el 96% de las aprobadas → si el objetivo es detectar todas las oportunidades de venta, este modelo es excelente.
- Tiene un F1-score muy alto en clase 1 (aprobada), ideal si el foco está en no perder clientes listos para cerrar.
- Altísimo número de falsos positivos: 116 ventas no aprobadas fueron clasificadas como si lo fueran.
- Recall de la clase 0 cae a 0.24 → el modelo casi no aprende a decir “esto va a ser rechazado”.

➔ Optimización de modelos ➔

Pruebo umbral de 0.6

```
# 1. Obtener probabilidades de clase 1
y_proba = best_xgb.predict_proba(X_test)[:, 1]

# 2. Aplicar nuevo umbral de decisión
y_pred_thresh = (y_proba > 0.6).astype(int)

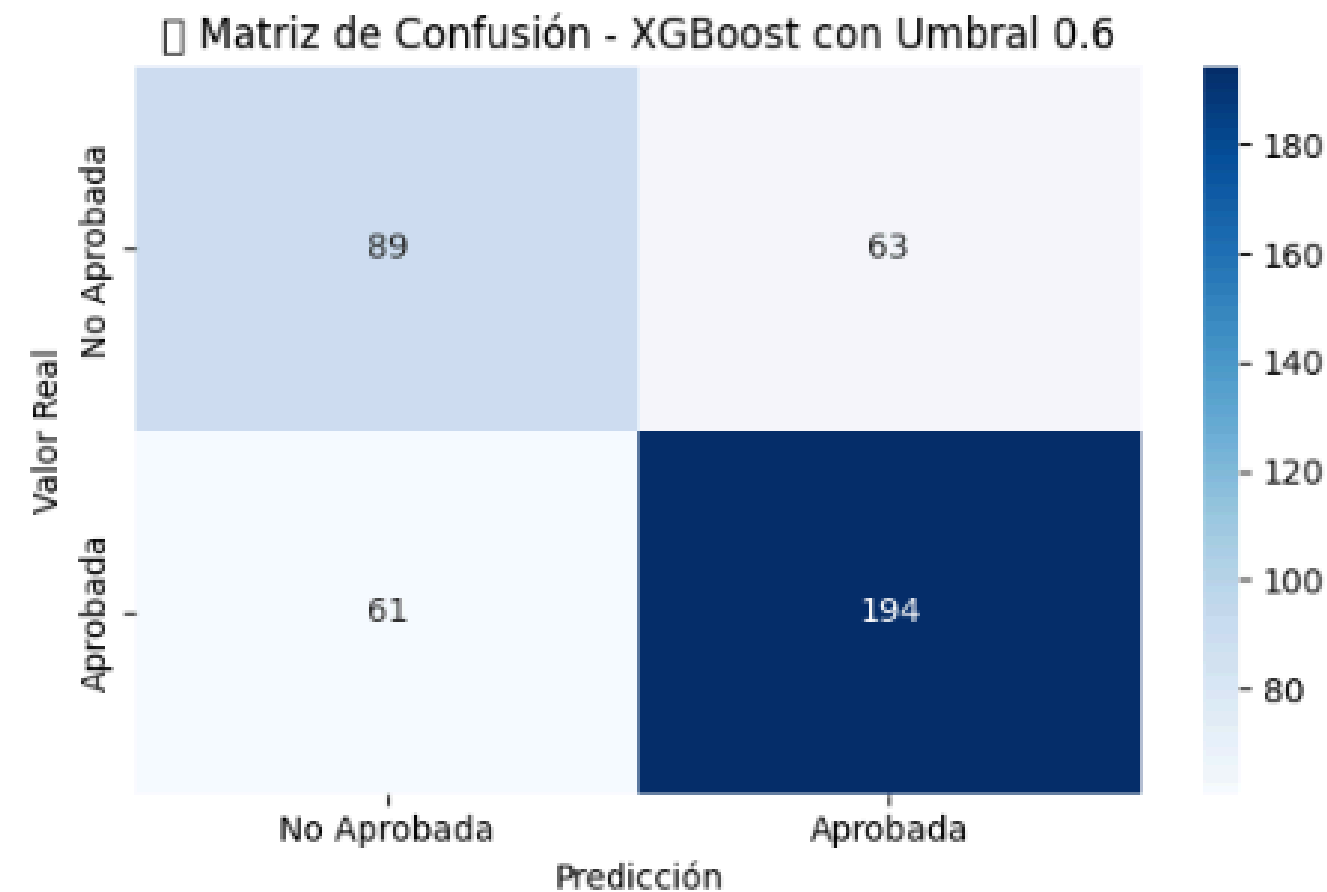
# 3. Matriz de confusión
cm = confusion_matrix(y_test, y_pred_thresh)
labels = ['No Aprobada', 'Aprobada']

plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels, yticklabels=labels)
plt.title("📊 Matriz de Confusión - XGBoost con Umbral 0.6")
plt.xlabel("Predicción")
plt.ylabel("Valor Real")
plt.tight_layout()
plt.show()

# 4. Reporte de métricas
print("\n📄 Reporte de Clasificación - Umbral 0.6:")
print(classification_report(y_test, y_pred_thresh))
```

➔ Optimización de modelos ➔

Pruebo umbral de 0.6



Reporte de Clasificación - Umbral 0.6:

	precision	recall	f1-score	support
0	0.59	0.59	0.59	152
1	0.75	0.76	0.76	255
accuracy			0.70	407
macro avg	0.67	0.67	0.67	407
weighted avg	0.69	0.70	0.69	407

Para mejorarlo aún más, ajuste un umbral a 0.6 y me dio como resultado:

Alta precisión y recall en aprobadas → no perdés tantas ventas

Mejor recuperación de las no aprobadas que con 0.5

F1-score muy sólido (0.76) para aprobadas

Mejor accuracy global (70%)

Es decir, es el más equilibrado.

CONCLUSIONES FINALES



Luego de desarrollar un modelo de machine learning capaz de predecir si una venta de portabilidad realizada por un call center será aprobada o no, utilizando datos históricos de clientes, producto, gestión comercial y características de los operadores.



Se construyeron y compararon distintos modelos (Random Forest y XGBoost), con un enfoque especial en:

- Limpieza y escalado de datos inteligente (usando StandardScaler para Edad y RobustScaler para las variables con outliers)
- Ingeniería de variables relevantes según correlación con el resultado
- Codificación adecuada de variables categóricas (Supervisor, Vendedor)
- Ajuste fino del umbral de clasificación, lo que permitió encontrar el mejor equilibrio entre recall y precisión



Modelo final seleccionado:

- XGBoost optimizado -

Con umbral ajustado a 0.6 para maximizar balance entre predicciones acertadas y reducción de errores

Resultado:

- **Detecta más del 76% de las ventas aprobadas**
- **Reduce significativamente los falsos positivos comparado con el umbral por defecto**
- **Ofrece un equilibrio realista para escenarios comerciales donde tanto la efectividad de ventas como la eficiencia operativa son importantes**





Conclusiones

El modelo desarrollado es robusto, balanceado y está alineado con los objetivos del negocio. Permite anticiparse al resultado de las ventas con una precisión sólida, brindando una herramienta valiosa para la gestión predictiva del rendimiento en campañas de portabilidad.