analisis_alivio_financiero

September 20, 2025

1 Análisis de campañas de alivio financiero y propuesta para recuperación de cartera vencida en una entidad financiera

2 *PROYECTO EN CONSTRUCCIÓN*

2.1 Breve descripción y alcance del proyecto

En el presente proyecto, se realiza el análisis de los datos correspondientes a una campaña que realiza la entidad financiera, a la cual se la referirá mediante el nombre ficticio ***Banco XYZ***, con el fin de recuperar al menos un 60% de su cartera vencida.

Este inconveniente le afecta al Banco dado que en el último semestre ha existido una baja de más del 15% en utilidades en comparación con el período anterior y se ha determinado mediante los reportes de auditoría interna que el **Departamento de Cobranzas** es uno de los que más fuertemente han contibuído a este déficit en las utilidades.

Una de las principales razones que generan las pérdidas en este departamento es la **Gestión de cartera vencida*, razón por la cual se realiza este análisis sobre las campañas de alivio financiero ofertadas previamente por el banco, llamadas* diferimiento* y *normalización*, mismas que han tenido baja efectividad en cuanto a recuperación monetaria frente a las metas previamente establecida.

Se van a responder a las siguientes preguntas para poder entender de mejor manera cuál es el problema actual, sus causas y de que manera se podría brindar una solución que satizfaga las necesidades del **Departamento de Cobranzas del Banco XYZ** sin descuidar el bienestar del cliente:

1. ¿Qué perfiles de clientes están siendo atendidos por cada campaña y canal?
2. ¿Cómo se distribuye la deuda según diferentes variables de análisis?
3. ¿Cómo podría optimizarse el enfoque actual para mejorar la efectividad de las campañas?

Para ofrecer una posible solución al **Departamento de Cobranzas** se va a centrar el enfoque en base a que tipos de perfiles tienen mayor deuda, por más tiempo, dónde se concentran más deudores demográficamente, si es que es necesario modificar las campañas actualmente ofrecidas por el Banco, ofrecer incentivos a los clientes para que se pongan al días con sus deudas de igual manera en base a determinados parámetros o variables, etc.

Nota: *A lo largo del presente reporte se analizan datos reales, mismos que han sido debidamente ofuscados y/u omitidos intencionalmente con el fin de proteger posible información sensible, así como por mantener respeto y apego hacia las leyes de protección de datos tanto locales como internacionales.*

2.2 Importación inicial de librerías y carga del dataset

Se importa la librería *pandas*, se carga el dataset y se muestran sus parámetros descriptivos incluyendo sus primeras filas, información general de tipos de datos, evaluación inicial de valores nulos, así como estadísticas descriptivas preliminares que otorga esta librería.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
df_morosidad = pd.read_csv("df_morosidad.csv")
```

2.3 Limpieza y preprocesamiento de la información

En este análisis se va a revisar la calidad y consistencia de la información de entrada, incluyendo la posible existencia de valores nulos o en blanco, datos atípicos, entre otros.

[3]:
```
df_morosidad.head()
```

[3]:
```
  codigo_cliente rango_morosidad  monto_credito  monto_por_vencer  \
0    DNI-9502992    31 - 60 DIAS         8000.0           5946.52
1    DNI-5110681    16 - 30 DIAS         5671.0              0.00
2    DNI-6527606     6 - 15 DIAS         5010.0              0.00
3    DNI-5095519    16 - 30 DIAS         5000.0              0.00
4    DNI-9856629    16 - 30 DIAS         6677.0           5511.13

   monto_vencido  intereses producto_host  dias_mora         gestor  \
0           0.00     143.94       PRECISO         36  RED COMERCIAL
1          94.16      98.97       PRECISO         22        DIGITAL
2          76.97     148.94       PRECISO         10        DIGITAL
3         146.04      65.76       PRECISO         23        DIGITAL
4           0.00     127.39       PRECISO         28  RED COMERCIAL

  recibe_sueldo_fijo provincia_cliente  region          sector_general
0                 SÍ         PICHINCHA  SIERRA          SECTOR PÚBLICO
1                 NO         PICHINCHA  SIERRA  TRANSPORTE Y LOGÍSTICA
2                 NO            GUAYAS   COSTA                COMERCIO
3                 NO         PICHINCHA  SIERRA                COMERCIO
4                 NO         PICHINCHA  SIERRA               SERVICIOS
```

[4]:
```
df_morosidad.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4115 entries, 0 to 4114
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   codigo_cliente      4115 non-null   object
```

```
 1   rango_morosidad     4115 non-null   object
 2   monto_credito       4115 non-null   float64
 3   monto_por_vencer    4115 non-null   float64
 4   monto_vencido       4115 non-null   float64
 5   intereses           4115 non-null   float64
 6   producto_host       4115 non-null   object
 7   dias_mora           4115 non-null   int64
 8   gestor              4115 non-null   object
 9   recibe_sueldo_fijo  4115 non-null   object
 10  provincia_cliente   4090 non-null   object
 11  region              4115 non-null   object
 12  sector_general      4115 non-null   object
dtypes: float64(4), int64(1), object(8)
memory usage: 418.1+ KB
```

[5]:
```
df_morosidad.describe()
```

[5]:
```
       monto_credito  monto_por_vencer  monto_vencido    intereses  \
count    4115.000000       4115.000000    4115.000000  4115.000000
mean     9878.836612       3740.186309      62.604493   165.204522
std     11170.864163       7052.096389     134.845050   181.270760
min      1132.000000          0.000000       0.000000     9.210000
25%      4449.500000          0.000000       0.000000    60.200000
50%      7000.000000       1481.000000       0.000000   113.110000
75%     11102.500000       5207.490000      89.695000   204.440000
max    205400.000000     136155.740000    3292.470000  2439.470000

         dias_mora
count  4115.000000
mean     26.249332
std      13.370148
min      10.000000
25%      15.000000
50%      23.000000
75%      35.000000
max      60.000000
```

2.3.1 Verificación de posibles valores blancos o nulos

[6]:
```
df_morosidad.isna().sum()
```

[6]:
```
codigo_cliente         0
rango_morosidad        0
monto_credito          0
monto_por_vencer       0
monto_vencido          0
intereses              0
```

```
producto_host          0
dias_mora              0
gestor                 0
recibe_sueldo_fijo     0
provincia_cliente     25
region                 0
sector_general         0
dtype: int64
```

Se puede ver en la celda anterior que de manera preliminar únicamente existen valores nulos en la variable *provincia_cliente* dentro del presente conjunto de datos, sin embargo si estos registros se eliminan se estaría subestimando la deuda que el Banco debe recuperar.

2.3.2 Verificación de posibles valores inconsistentes

Se realiza un análisis de los valores únicos por cada columna para determinar si podría resultar conveniente convertirlos al tipo categórico de pandas.

[7]:
```
df_morosidad.dtypes
```

[7]:
```
codigo_cliente         object
rango_morosidad        object
monto_credito         float64
monto_por_vencer      float64
monto_vencido         float64
intereses             float64
producto_host          object
dias_mora               int64
gestor                 object
recibe_sueldo_fijo     object
provincia_cliente      object
region                 object
sector_general         object
dtype: object
```

[8]:
```
condiciones = [
    (df_morosidad["dias_mora"] >= 10) & (df_morosidad["dias_mora"] <= 20),
    (df_morosidad["dias_mora"] >= 21) & (df_morosidad["dias_mora"] <= 60)
]

resultados = ["NORMALIZACIÓN", "DIFERIMIENTO"]
df_morosidad["tipo_campania"] = np.select(condiciones, resultados,␣
 ↪default="NO_APLICA")
```

[9]:
```
posibles_columnas_categoricas = [
    "rango_morosidad",
    "producto_host",
    "gestor",
```

```
    "recibe_sueldo_fijo",
    "provincia_cliente",
    "region",
    "sector_general",
    "tipo_campania"
]

for col in posibles_columnas_categoricas:
    print(f'Valores únicos en "{col}":')
    print(df_morosidad[col].unique())
    print("-" * 50)
```

```
Valores únicos en "rango_morosidad":
['31 - 60 DIAS' '16 - 30 DIAS' '6 - 15 DIAS']
--------------------------------------------------
Valores únicos en "producto_host":
['PRECISO' 'AUTOS' 'PRODUCTO MUJER' 'PRODUCTIVO' 'MICROFINANZAS' 'HABITAR']
--------------------------------------------------
Valores únicos en "gestor":
['RED COMERCIAL' 'DIGITAL']
--------------------------------------------------
Valores únicos en "recibe_sueldo_fijo":
['SÍ' 'NO']
--------------------------------------------------
Valores únicos en "provincia_cliente":
['PICHINCHA' 'GUAYAS' 'LOS RIOS' 'SANTO DOMINGO DE LOS TSACHILAS'
 'SUCUMBIOS' 'MANABI' nan 'EL ORO' 'CHIMBORAZO' 'AZUAY' 'ORELLANA' 'NAPO'
 'ESMERALDAS' 'TUNGURAHUA' 'ZAMORA CHINCHIPE' 'PASTAZA' 'GALAPAGOS'
 'BOLIVAR' 'IMBABURA' 'LOJA' 'COTOPAXI' 'SANTA ELENA' 'CAÑAR'
 'MORONA SANTIAGO' 'CARCHI']
--------------------------------------------------
Valores únicos en "region":
['SIERRA' 'COSTA' 'AMAZONIA' 'NO_ESPECIFICADO' 'GALAPAGOS']
--------------------------------------------------
Valores únicos en "sector_general":
['SECTOR PÚBLICO' 'TRANSPORTE Y LOGÍSTICA' 'COMERCIO' 'SERVICIOS'
 'INDUSTRIA / MANUFACTURA' 'MINERÍA Y EXTRACCIÓN'
 'SIN ACTIVIDAD ECONÓMICA' 'EDUCACIÓN Y SALUD' 'INDUSTRIA'
 'SECTOR FINANCIERO' 'AGROPECUARIO' 'CONSTRUCCIÓN'
 'TECNOLOGÍA Y TELECOMUNICACIONES']
--------------------------------------------------
Valores únicos en "tipo_campania":
['DIFERIMIENTO' 'NORMALIZACIÓN']
--------------------------------------------------
```

Se puede observar que no existen valores inconsistentes al haber analizado los valores únicos en cada columna las columnas o variables ***rango_morosidad***, ***producto_host***, ***gestor***, ***recibe_sueldo_fijo***, ***region*** y ***sector_general*** pueden convertirse a un tipo de variable

categórico. Además la variable ***recibe_sueldo_fijo*** puede convertirse a binaria para un procesamiento más eficiente.

[10]:
```
df_morosidad[posibles_columnas_categoricas] =␣
 ↪df_morosidad[posibles_columnas_categoricas].astype("category")
df_morosidad["recibe_sueldo_fijo"] = df_morosidad["recibe_sueldo_fijo"]\
    .map({"SI": True, "NO": False}).astype("bool")
```

[11]:
```
df_morosidad.dtypes
```

[11]:
```
codigo_cliente          object
rango_morosidad       category
monto_credito          float64
monto_por_vencer       float64
monto_vencido          float64
intereses              float64
producto_host         category
dias_mora                int64
gestor                category
recibe_sueldo_fijo        bool
provincia_cliente     category
region                category
sector_general        category
tipo_campania         category
dtype: object
```

Se verifican también los valores mínimos y máximos con el fin de entender cuáles de los valores no pueden ser igual a cero (0).

Valores numéricos mínimos:

[12]:
```
df_morosidad.select_dtypes(include="number").min().apply(lambda x: f"{x:,.2f}")
```

[12]:
```
monto_credito       1,132.00
monto_por_vencer        0.00
monto_vencido           0.00
intereses               9.21
dias_mora              10.00
dtype: object
```

Valores numéricos máximos:

[13]:
```
df_morosidad.select_dtypes(include="number").max().apply(lambda x: f"{x:,.2f}")
```

[13]:
```
monto_credito       205,400.00
monto_por_vencer    136,155.74
monto_vencido         3,292.47
intereses             2,439.47
dias_mora                60.00
```

```
dtype: object
```

Diferencia absoluta entre monto vencido y monto por vencer:

[14]:
```
np.min(np.abs(df_morosidad["monto_por_vencer"] - df_morosidad["monto_vencido"]))
```

[14]:
```
0.3
```

Se observan algunos apectos relevantes en cuanto a estos datos, el primero es que el rango de días en mora en el *dataset* está comprendido entre 10 y 60 días, mientras que los montos por vencer y montos vencidos pueden tener valores en cero (0), sin embargo la resta de ambos jamás puede ser cero (0), esto debido a que en el presente conjunto de datos únicamente se contemplan a clientes que cuenten todavía con saldos o valores pendientes en sus préstamos a la fecha de corte del conjunto de datos.

A continuación se analizará si es que existen clientes que tengan al mismo tiempo *montos vencidos* y *montos por vencer* y de ser así se determinará si se trata o no de posibles inconsistencias realizando análisis más exhaustivos de ser el caso.

[15]:
```
len(df_morosidad.query("monto_por_vencer != 0 and monto_vencido != 0"))
```

[15]:
```
0
```

[16]:
```
print(len(df_morosidad.query("monto_vencido == 0")))
print(len(df_morosidad.query("monto_por_vencer == 0")))

print()

print(len(df_morosidad.query("tipo_campania == 'DIFERIMIENTO'")))
print(len(df_morosidad.query("tipo_campania == 'NORMALIZACIÓN'")))
```

```
2251
1864

2340
1775
```

[17]:
```
resaltado = pd.DataFrame({
    'monto_por_vencer_resaltado': [
        len(df_morosidad.query("tipo_campania == 'DIFERIMIENTO' &␣
↪monto_por_vencer > 0")),
        len(df_morosidad.query("tipo_campania == 'NORMALIZACIÓN' &␣
↪monto_por_vencer > 0"))
    ],
    'monto_vencido_resaltado': [
        len(df_morosidad.query("tipo_campania == 'DIFERIMIENTO' & monto_vencido␣
↪> 0")),
```

```
        len(df_morosidad.query("tipo_campania == 'NORMALIZACIÓN' &␣
 ↪monto_vencido > 0"))
    ]
}, index=['DIFERIMIENTO', 'NORMALIZACIÓN'])

resaltado
```

[17]:

	monto_por_vencer_resaltado	monto_vencido_resaltado
DIFERIMIENTO	1266	1074
NORMALIZACIÓN	985	790

[18]:
```
total_normalizacion = df_morosidad.query("tipo_campania ==␣
 ↪'NORMALIZACIÓN'")["monto_credito"].sum()
total_diferimiento = df_morosidad.query("tipo_campania ==␣
 ↪'DIFERIMIENTO'")["monto_credito"].sum()

print(f"Total monto_credito NORMALIZACIÓN:\t{total_normalizacion:,.2f}")
print(f"Total monto_credito DIFERIMIENTO:\t{total_diferimiento:,.2f}")
```

```
Total monto_credito NORMALIZACIÓN:      16,884,500.90
Total monto_credito DIFERIMIENTO:       23,766,911.76
```

[19]:
```
print(len(df_morosidad))
```

```
4115
```

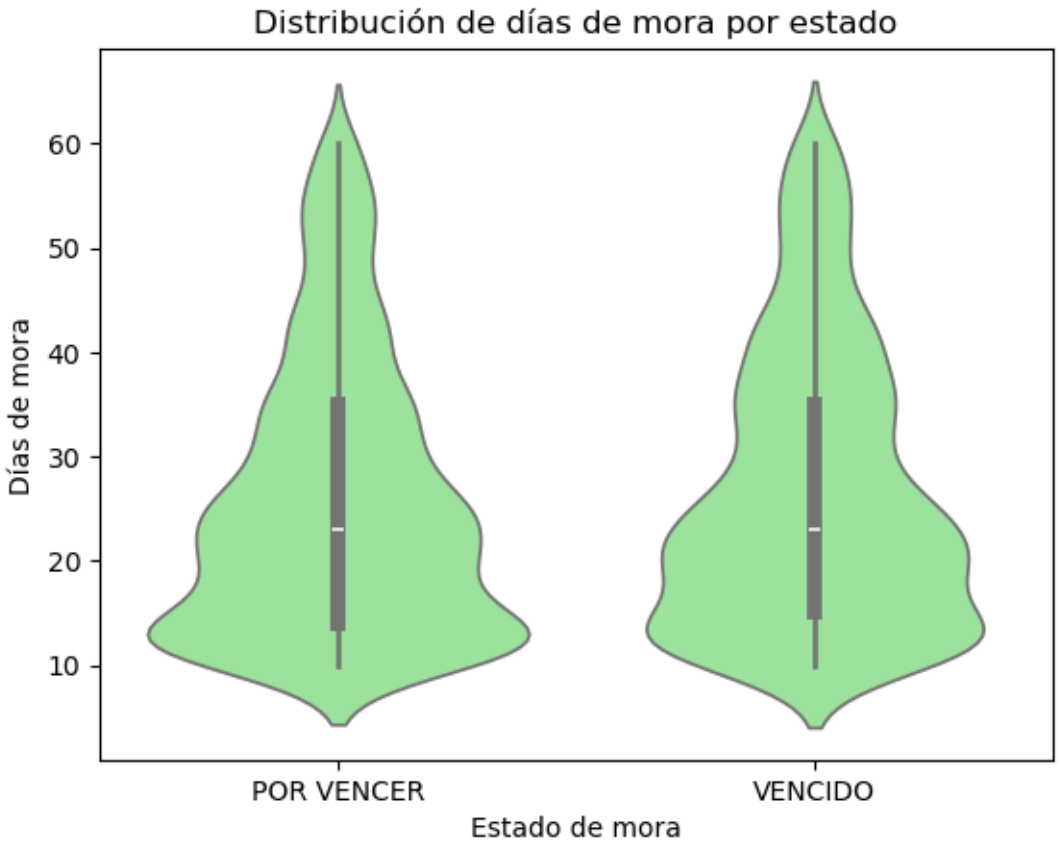
Como se observa en la celda anterior, no existen registros (*clientes*) con *montos por vencer* y al mismo tiempo *montos vencidos*, lo cual indica de manera preliminar que los clientes con *montos por vencer* podrían ser clientes que todavía no caen en **mora** mientras que los clientes con *montos vencidos* serían aquellos que ya están en mora, sin emabrgo al ser este un conjunto de datos que trata sobre clientes en mora y haberse demostrado previamente que el rango de mora para todos y cada uno de los clientes se encuentra entre **10 y 60 días**, este relato no tendría sentido.

Con el objetivo de ampliar el análisis se procederá a graficar la distribución de *días en mora* para entender su relación con los clientes que están en estado **POR VENCER** o **VENCIDO**. Para ello, en primer lugar se agregará una categoría llamada **estado_mora** con los estados mencionados previamente con el fin de simplificar el análisis.

[20]:
```
df_morosidad["estado_mora"] = np.where(
    df_morosidad["monto_por_vencer"] != 0, "POR VENCER", "VENCIDO"
)
```

[21]:
```
sns.violinplot(
    data=df_morosidad,
    x="estado_mora",
    y="dias_mora",
    color="lightgreen"
)
```

```
plt.title("Distribución de días de mora por estado")
plt.xlabel("Estado de mora")
plt.ylabel("Días de mora")
plt.show()
```



El análisis del boxplot muestra que las distribuciones de días en mora son muy similares entre los clientes con montos "por vencer" y aquellos con montos "vencidos", tanto a nivel de mediana como de cuartiles. Esto indica que el hecho de que uno de los valores (monto_vencido o monto_por_vencer) sea cero no significa que el cliente no esté en mora, ni que la otra columna sea realmente cero. Más bien, refleja que el banco segmenta los clientes para resaltar en algunos casos los montos por vencer y en otros los montos vencidos.

A continuación, se procederá a calcular el porcentaje tanto de los montos por vencer como de los montos vencidos para detectar patrones y continuar con la investigación sobre el porqué el banco busca resaltar un monto u otro.

[22]:
```
df_morosidad["pct_vencido"] = (df_morosidad["monto_vencido"] /␣
  ↪df_morosidad["monto_credito"]).round(4)
df_morosidad["pct_por_vencer"] = (df_morosidad["monto_por_vencer"] /␣
  ↪df_morosidad["monto_credito"]).round(4)
```

[23]:
```
print("Máximos y mínimos (respectivamente) en 'pct_por_vencer'")
print(df_morosidad[df_morosidad["pct_por_vencer"] != 0]["pct_por_vencer"].
 ↪max()*100)
print(df_morosidad[df_morosidad["pct_por_vencer"] != 0]["pct_por_vencer"].
 ↪min()*100)

print("Máximos y mínimos (respectivamente) en 'pct_vencido'")
print(df_morosidad[df_morosidad["pct_vencido"] != 0]["pct_vencido"].max()*100)
print(df_morosidad[df_morosidad["pct_vencido"] != 0]["pct_vencido"].min()*100)
```

```
Máximos y mínimos (respectivamente) en 'pct_por_vencer'
98.97
8.9
Máximos y mínimos (respectivamente) en 'pct_vencido'
8.52
0.01
```

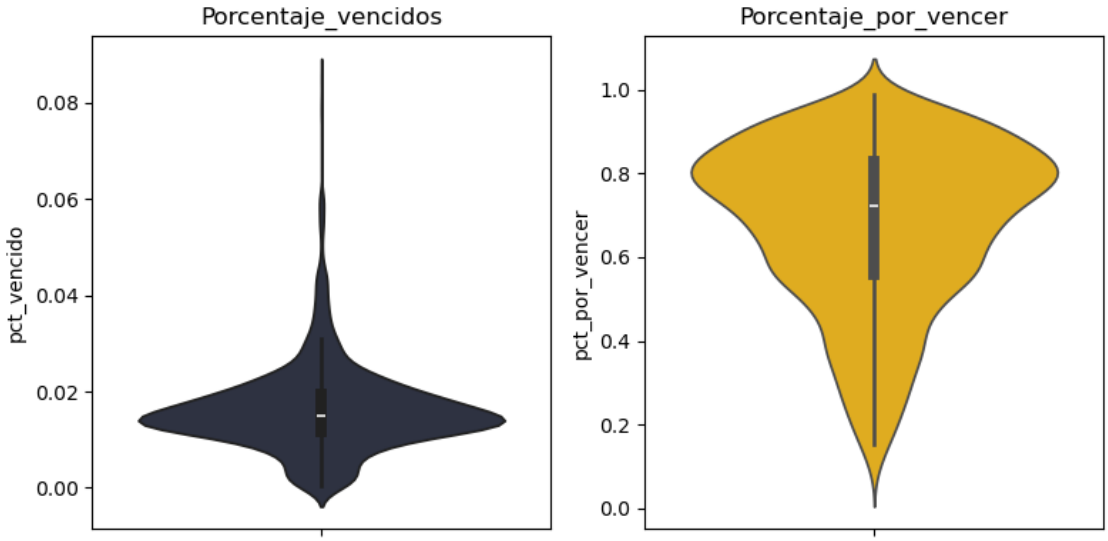
Como se puede ver en los porcentajes de montos por vencer los valores van entre el 8.9% y el 98.97% del total del crédito como máximo, mientras que en los porcentajes de los montos vencidos los mínimos y máximos oscilan entre 0.01% y 8.52%, evidenciándose continuidad entre ambos grupos de porcentajes, por lo cual todo cliente tiene una parte vencida pequeña y una parte por vencer grande, y el sistema decide cual resaltar.

[24]:
```
fig, axes = plt.subplots(1, 2, figsize=(8, 4))

sns.violinplot(
    data=df_morosidad.query("pct_vencido != 0"),
    y="pct_vencido",
    ax=axes[0],
    color="#2b3044"
)
axes[0].set_title("Porcentaje_vencidos")

sns.violinplot(
    data=df_morosidad.query("pct_por_vencer != 0"),
    y="pct_por_vencer",
    ax=axes[1],
    color="#ffbb00"
)
axes[1].set_title("Porcentaje_por_vencer")

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



Para proseguir con el análisis, se va a indagar sobre las posibles causas que hacen que el banco resalte en algunos casos solo **monto_por_vencer** o solo **monto_vencido**, para definir el enfoque posterior en este análisis.

En primer lugar se van a contar los valores por cada una de las posibles categorías que influyan contra el tipo de estado de mora, sea este por vencer o vencido.

[25]:
```
pd.crosstab(df_morosidad["producto_host"], df_morosidad["estado_mora"])\
    .reindex(columns=["POR VENCER", "VENCIDO"], fill_value=0) \
    .sort_index()
```

[25]:
```
estado_mora     POR VENCER  VENCIDO
producto_host
AUTOS                   90       77
HABITAR                 18        0
MICROFINANZAS          903      394
PRECISO               1179     1372
PRODUCTIVO              40       14
PRODUCTO MUJER          21        7
```

[26]:
```
pd.crosstab(df_morosidad["rango_morosidad"], df_morosidad["estado_mora"])\
    .reindex(columns=["POR VENCER", "VENCIDO"], fill_value=0) \
    .sort_index()
```

[26]:
```
estado_mora      POR VENCER  VENCIDO
rango_morosidad
16 - 30 DIAS            890      753
31 - 60 DIAS            688      620
6 - 15 DIAS             673      491
```

[27]:
```
pd.crosstab(df_morosidad["gestor"], df_morosidad["estado_mora"])\
    .reindex(columns=["POR VENCER", "VENCIDO"], fill_value=0) \
    .sort_index()
```

[27]:
```
estado_mora    POR VENCER  VENCIDO
gestor
DIGITAL              1892     1572
RED COMERCIAL         359      292
```

[28]:
```
pd.crosstab(df_morosidad["recibe_sueldo_fijo"], df_morosidad["estado_mora"])\
    .reindex(columns=["POR VENCER", "VENCIDO"], fill_value=0) \
    .sort_index()
```

[28]:
```
estado_mora         POR VENCER  VENCIDO
recibe_sueldo_fijo
False                     1107      945
True                      1144      919
```

[52]:
```
pd.crosstab(df_morosidad["region"], df_morosidad["estado_mora"])\
    .reindex(columns=["POR VENCER", "VENCIDO"], fill_value=0) \
    .sort_index()
```

[52]:
```
estado_mora      POR VENCER  VENCIDO
region
AMAZONIA                125      104
COSTA                   961      813
GALAPAGOS                11        4
NO_ESPECIFICADO          12       13
SIERRA                 1142      930
```

[30]:
```
pd.crosstab(df_morosidad["sector_general"], df_morosidad["estado_mora"])\
    .reindex(columns=["POR VENCER", "VENCIDO"], fill_value=0) \
    .sort_index()
```

[30]:
```
estado_mora                    POR VENCER  VENCIDO
sector_general
AGROPECUARIO                          242      171
COMERCIO                              722      514
CONSTRUCCIÓN                           34       39
EDUCACIÓN Y SALUD                     134      139
INDUSTRIA                              49       41
INDUSTRIA / MANUFACTURA               141       93
MINERÍA Y EXTRACCIÓN                   18       15
SECTOR FINANCIERO                      18       21
SECTOR PÚBLICO                        104       57
SERVICIOS                             474      456
SIN ACTIVIDAD ECONÓMICA               233      232
```

```
TECNOLOGÍA Y TELECOMUNICACIONES          25       23
TRANSPORTE Y LOGÍSTICA                   57       63
```

Hasta el momento en base al análisis de la relación de estas variables categóricas con lo que el banco quiere resltar en cada cliente sus montos por vencer o vencidos se tiene lo siguiente:

rango_morosidad: Distribuciones muy parecidas entre "POR VENCER" y "VENCIDO" (ligero sesgo: 31–60 días pesa un poco más en VENCIDO y 6–15 en POR VENCER). No parece ser el driver.

gestor: DIGITAL domina en ambos (85% en ambos estados). Casi sin diferencia → no explica el "resaltado".

recibe_sueldo_fijo: Proporciones casi idénticas (leve sesgo hacia "False" en VENCIDO). Efecto menor.

provincia: Pichincha/Guayas concentran clientes en ambos estados; sin normalizar por base/penetración regional, no se infiere causalidad. Probable efecto de tamaño de mercado.

sector_general: Diferencias pequeñas (p. ej., Servicios y Sin actividad algo más en VENCIDO; Sector público y Agropecuario algo más en POR VENCER). No parece fuerte.

producto_host: Sí muestra asociación real (Cramér's V 0.23, p 0.001). Es el único con señal clara.

Se va a realizar una análisis adicional con chi-cuadrado para ayudar a corroborar o falsar la hipótesis de que el Banco podría estar eligiendo qué tipo de monto resaltar por cada cliente en base al ***producto_host***.

[32]:
```
def cramers_v(ct):
    chi2, p, dof, exp = chi2_contingency(ct)
    n = ct.to_numpy().sum()
    k = min(ct.shape)
    return p, np.sqrt(chi2 / (n * (k - 1)))

targets = ["producto_host","rango_morosidad","gestor",
           "recibe_sueldo_fijo","provincia_cliente","sector_general"]

res = []
for col in targets:
    ct = pd.crosstab(df_morosidad[col], df_morosidad["estado_mora"])
    p, v = cramers_v(ct)
    res.append((col, p, v))
pd.DataFrame(res, columns=["variable","p_value","cramers_v"]).
 ↪sort_values("cramers_v", ascending=False)
```

[32]:

	variable	p_value	cramers_v
0	producto_host	3.235350e-45	0.230390
4	provincia_cliente	7.022449e-05	0.119245
5	sector_general	2.159893e-04	0.094939
1	rango_morosidad	2.897559e-02	0.041487

```
3  recibe_sueldo_fijo  3.477512e-01   0.014637
2              gestor  8.376312e-01   0.003195
```

El análisis chi-cuadrado muestra una relación estadísticamente significativa entre el tipo de producto (producto_host) y el estado resaltado (POR VENCER vs VENCIDO), con un tamaño de efecto no trivial (Cramér's V = 0.230; p 0.001), lo que sugiere que la decisión de resaltar uno u otro componente depende principalmente de las características del producto. Otras variables presentan asociaciones mucho menores: la provincia y el sector tienen efectos débiles (V 0.12 y V 0.095, respectivamente), el rango de morosidad apenas aporta señal (V 0.041) y ni el canal de gestión ni la condición de sueldo fijo muestran evidencia relevante.

Por ende los resultados indican una orientación más fuerte hacia el *perfil del producto* o *producto_host*, en lugar de la *antigüedad exacta del atraso* o el *canal*.

[ ]:

2.3.3 Clientes por tipo de campaña

[34]:
```
# plt.figure(figsize=(4, 3))
# sns.countplot(
#     data=df_morosidad,
#     x="tipo_campania"
# )

# plt.title("Clientes por tipo de campaña")
# plt.ylabel("Cantidad de clientes")
# plt.xlabel("Campaña")
# plt.show()
```

[35]:
```
# plt.figure(figsize=(4, 3))
# sns.countplot(
#     data=df_morosidad,
#     x="tipo_campania"
# )

# plt.title("Clientes por tipo de campaña")
# plt.ylabel("Cantidad de clientes")
# plt.xlabel("Campaña")
# plt.show()
```

2.3.4 Análisis por Canal de Gestión

[36]:
```
# plt.figure(figsize=(5.5, 3.5))

# sns.countplot(
#     data=df_morosidad,
#     x="tipo_campania",
```

```
#     hue="gestor"
# )

# plt.title("Clientes por campaña y canal de atención")
# plt.ylabel("Cantidad de clientes")
# plt.xlabel("Campaña")
# plt.legend(title="Canal")
# plt.show()
```

[37]:
```
# sns.countplot(
#     data=df_morosidad,
#     x="tipo_campania",
#     hue="recibe_sueldo_fijo"
# )
# plt.title("Clientes con/sin sueldo fijo por campaña")
# plt.ylabel("Cantidad de clientes")
# plt.xlabel("Campaña")
# plt.legend(title="Recibe sueldo fijo")
# plt.show()
```

[38]:
```
# sns.scatterplot(data=df_morosidad, x="dias_mora", y="monto_vencido",␣
 ↪hue="tipo_campania")
# plt.title("Días de mora vs Monto vencido")
# plt.xlabel("Días en mora")
# plt.ylabel("Monto vencido ($)")
# plt.legend(title="Campaña")
# plt.show()
```

Se puede observar que en la campaña de **DIFERIMIENTO** predominan deudas bajas entre USD 100 y USD 500 aproximadamente, con variabilidad relativamente baja al existir pocos valores extremos.

Por otro lado, en el caso de la campaña de **NORMALIZACIÓN** el monto vencido promedio se muestra ligeramente más alto, con una mayor presencia de valores atípicos, rondando en algunos casos valores por encima de USD 1000 y en ocasiones incluso USD 3000.

Se puede ver entonces de manera preliminar que estos clientes representan un riesgo de impago más alto para el banco, por lo cual se podría considerar para la campaña de **NORMALIZACIÓN** plantear una estrategia correctiva mientras que en **DIFERIMIENTO** podría ser todavía una estrategia preventiva.

[39]:
```
# provincia_counts = df_morosidad["region"].value_counts().sort_values()
# provincia_counts.plot(kind="barh", figsize=(8,6))
# plt.title("Clientes por región")
# plt.xlabel("Cantidad de clientes")
# plt.ylabel("Región")
# plt.show()
```

[40]:
```
# provincia_counts = df_morosidad["provincia_cliente"].value_counts().
 ↪sort_values()\
# .nlargest(6)
# provincia_counts.plot(kind="barh", figsize=(5,3))
# plt.title("Clientes por región")
# plt.xlabel("Cantidad de clientes")
# plt.ylabel("Provincia")
# plt.show()
```

[41]:
```
# plt.figure(figsize=(10,6))
# sns.countplot(
#     data=df_morosidad,
#     y="sector_general",
#     hue="tipo_campania",
#     order=df_morosidad["sector_general"].value_counts().index
# )
# plt.title("Clientes por sector económico y campaña")
# plt.xlabel("Cantidad de clientes")
# plt.ylabel("Sector económico")
# plt.legend(title="Campaña")
# plt.show()
```

[42]:
```
# print("Total de clientes:\t\t", len(df_morosidad))
# print(f"Total monto vencido: \t\t${df_morosidad['monto_vencido'].sum():,.2f}")
# print(f"Total intereses acumulados: \t${df_morosidad['intereses'].sum():,.
 ↪2f}")
# print("Clientes con sueldo fijo:\t",␣
 ↪df_morosidad[df_morosidad["recibe_sueldo_fijo"] == True].shape[0])
```

[43]:
```
# df_morosidad[['monto_credito', 'monto_por_vencer', 'monto_vencido']].sum() \
# .apply(lambda x: f"$ {x:,.2f}")
```

[44]:
```
# df_morosidad[['monto_credito', 'monto_por_vencer', 'monto_vencido',␣
 ↪'intereses']].sum() \
# .plot(kind= "bar")

# plt.show()
```

[45]:
```
# fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# # 1er gráfico: monto_credito vs intereses
# axes[0].scatter(df_morosidad["monto_credito"], df_morosidad["intereses"])
# axes[0].set_title("Monto crédito vs Intereses")
# axes[0].set_xlabel("Monto del crédito")
# axes[0].set_ylabel("Intereses")

# # 2do gráfico: monto_por_vencer vs intereses
```

```
# axes[1].scatter(df_morosidad["monto_por_vencer"], df_morosidad["intereses"],␣
 ↪color="orange")
# axes[1].set_title("Monto por vencer vs Intereses")
# axes[1].set_xlabel("Monto por vencer")
# axes[1].set_ylabel("Intereses")

# # 3er gráfico: monto_vencido vs intereses
# axes[2].scatter(df_morosidad["monto_vencido"], df_morosidad["intereses"],␣
 ↪color="green")
# axes[2].set_title("Monto vencido vs Intereses")
# axes[2].set_xlabel("Monto vencido")
# axes[2].set_ylabel("Intereses")

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

[46]:
```
# plt.scatter(df_morosidad["dias_mora"], df_morosidad["intereses"])
# plt.title("Scatter Días Mora vs. Intereses")
# plt.xlabel("Días Mora")
# plt.ylabel("Intereses")
# plt.show()
```

[47]:
```
# df_morosidad.groupby(["tipo_campania"], observed=True)\
#     [['monto_credito', 'monto_por_vencer', 'monto_vencido', 'intereses']].
 ↪sum()
```

[48]:
```
# df_morosidad[['monto_vencido', 'intereses']].sum() \
# .plot(kind= "bar")

# plt.show()
```

[49]:
```
# df_morosidad[['monto_credito', 'monto_por_vencer', 'monto_vencido',␣
 ↪'dias_mora']].head(10)
```

[50]:
```
# df_morosidad[['monto_credito', 'monto_por_vencer', 'monto_vencido',␣
 ↪'dias_mora']].describe()
```

[51]:
```
# (df_morosidad['monto_vencido'] / df_morosidad['monto_credito']).describe()
```

[ ]:
```

```