Modelación: Calibración y Predicción

Desenlace: Cáncer de Mama

Autor(es):

- **Nombre:** Juan Andrés Echeverri
  Cargo: Científico de Datos Datalytics
  Correo: juan.echeverri@datalytics.com

Descripción:

- El siguiente libro, representa una plantilla para la ejecución de un análisis supervisado de clasificación. Está compuesto por una secuencia de pasos, que van desde la carga de archivos con funciones pre-diseñadas para realizar los distintos análisis y estimaciones, hasta el almacenamiento del objeto con el modelo entrenado y demás componentes necesarios para un proceso de producción. Cada una de estas fases se encuentran explicadas y documentadas a lo largo de la plantilla.

  > *Parte de los procedimientos aquí contenidos, fueron pensados en un escenario de variable de respuesta dicotómica, como la mayor cantidad de adelantos metodológicos en esta materia.*

---

1. Contexto general del problema

En el contexto de gestión del riesgo individual en salud, se desea desarrollar un modelo de clasificación, para predecir la presencia de **cáncer de mama**, a partir de población general, individuo a individuo. Posteriormente, se construirá un Índice de Salud, a partir de la estratificación del riesgo de cada paciente y la unificación de los resultados de los modelos para distintos desenlaces *(priorizados previamente)*. Para un mayor detalle del problema, se abordan los siguientes puntos:

- **Desenlace**: Insuficiencia Cáncer de Mama

- **Definición de la población**:
  - **Enfermos:** Mujeres mayores de 18 años, marcados o con diagnóstico de cáncer de mama, en el período muestral
  - **Sanos:** Mujeres mayores de 18 años, sin marca o sin diagnóstico de cáncer de mama, en el período muestral

- **Fuentes de definición**: Sistema de marcación, diagnósticos de algunas consultas médicas y prestaciones determinadas

- **Fecha del evento**:
  - **Enfermos:** Fecha de inicio de vigencia de la marca confirmada, o la primera fecha de diagnóstico
  - **Sanos:** Un año antes del límite superior del período muestral

- **Período de pronóstico**: 1 año

- **Unidad de observación**: Individuo

- **Tipo de estudio**: Cohorte pseudo-transversal

- **Período muestral**: Enero 2012 - Enero 2019

- **Negocios involucrados**: Seguros Obligatorio y Voluntario en Salud

---

2. Carga de funciones

2.1. Manejo de rutas

2.1.1. Definición

Definición de las rutas que serán necesarias para cargar funciones creadas por el usuario y que reposan en rutas que no están definido por defecto en el sistema.

In [1]:
```
# Lista con las rutas relativas, necesarias para la carga de funciones propias
req_paths = ['../scripts/']
```

2.1.2. Adecuación

Transformación de la lista de rutas que trae por defecto el sistema, para que la búsqueda de paquetes y funciones propias se inicie en las rutas modificadas.

In [2]:
```
# Provee acceso a funciones y objetos mantenidos por del intérprete
import sys
# Transformación de la ruta general
sys.path = req_paths + sys.path
```

2.2. Carga

2.2.1. Propias

Carga las funciones propias para ejecutar el proceso de modelación.

In [3]:
```
from cells_format import remove_scroll_output
from data_connections import (load_data, save_data, save_object)
from corr_analysis import (print_corr_vars_selection, print_multicol_analysis)
from data_definition import (DependEncodedData, EncodedDataParams, ModelEncodedData, ModelTransformerData, ModelDataParams,
                             init_default_categories, non_excluded_data)
from indiv_logistic_effects import (print_detailed_imbalance_table, print_bivar_infer_analysis)
from missing_data import (print_null_count_analysis, print_cond_null_analysis)
from specification import (FeaturesSelectors, LogisticRegressionModelling, print_logistic_results)
from training import ClassificationTrainingValidation
from validation import (calibration_curves, cv_scoring_graphs, learning_curves)
from bestmodel_selection import scoring_best_models
from cutoff_selection import (print_cut_offs_selection, opt_cutoff_class_report, _y_pred_prediction_method)
from risk_stratification import (IndividualRiskStratification, print_risk_stratification)
from imputation import ImputationMissingValues
```

2.2.2. Auxiliares

Carga las funciones auxiliares para ejecutar el proceso de modelación.

In [4]:
```
import numpy as np
import pandas as pd
from patsylearn import PatsyTransformer
from sklearn.model_selection import (StratifiedKFold, RepeatedStratifiedKFold, train_test_split)
from sklearn.impute import SimpleImputer
import ipywidgets as widgets
import time
```

---

3. Carga de datos

3.1. Definición de parámetros

Parámetros necesarios para la carga de información, derivada del proceso del análisis descriptivo.

In [5]:
```
# Ruta relativa, utilizada para almacenar la plantilla generada
data_path = '../data/'
# Nombre de la base de datos generada
data_file = 'data_cam'
#data_file = 'cm4'
data_ant_cancer = 'ant_fam_cancer_otros'
# Formato de almacenamiento de la base de datos. Los formatos disponibles, son: ["parquet", "pickle", "txt", "csv", "json",
data_fmt = 'parquet'
#data_fmt = 'csv'
# Separador de la base de datos, cuando el formato de almacenamiento está en la siguiente lista: ["txt", "csv"]
data_sep = None
```

3.2. Ejecución

Carga la base de datos principal.

In [6]:
```
df = load_data(data_path=data_path, data_name=data_file, fmt=data_fmt, sep=data_sep)
df = df[df['Edad'].between(30,75)]
#df = pd.read_csv(data_path+data_file+'.'+data_fmt)
#df_ant = pd.read_csv(data_path+data_ant_cancer+'.'+data_fmt,sep = ';')
```

In [7]:
```
#df['Ind_Frecuencia_Cigarrillo'].value_counts()
```

In [8]:
```
#df[df['Ind_Frecuencia_Cigarrillo'].isna()]
```

In [9]:
```
df.columns
```

Out[9]:
```
Index(['Afiliado_Id', 'Asegurado_Id', 'Dni_Cliente', 'Fecha_Corte', 'Dias_EPS',
       'Dias_SEG', 'Ind_Mayoria_Licor', 'Ind_Frecuencia_Licor', 'Sexo_Cd',
       'Raza_Desc', 'Valor_IMC', 'Num_Edad_Menopausia', 'Num_Edad_Menarca',
       'Ind_Terapia_Hormonal', 'Num_Birads', 'Ind_Ooforectomia_Bilateral',
       'Num_Fam_Primer_Grado_Otros', 'Num_Fam_Segundo_Grado_Otros',
       'Ind_Ant_Fam_Otros', 'Num_Fam_Primer_Grado_CAM',
       'Num_Fam_Segundo_Grado_CAM', 'Ind_Ant_Fam_CAM', 'Ind_Ant_Radio_Torax',
       'Fecha_Nacimiento', 'Edad', 'Ind_CAM'],
      dtype='object')
```

In [10]:
```
conditions = [
    (df['Dias_EPS'] > 0) & (df['Dias_SEG'] == 0),
    (df['Dias_EPS'] == 0) & (df['Dias_SEG'] > 0),
    (df['Dias_EPS'] > 0) & (df['Dias_SEG'] > 0)
]

values = ['Obligatorios', 'Voluntarios', 'Obligatorios y Voluntarios']

# Aplicar las condiciones y asignar valores a una nueva columna 'Tipo_Dias'
df['Ind_Negocio'] = np.select(conditions, values)
df['Ind_Negocio'].value_counts()
```

Out[10]:
```
Obligatorios                1321647
Obligatorios y Voluntarios     215791
Voluntarios                     13672
Name: Ind_Negocio, dtype: int64
```

In [11]:
```
df = df[df['Ind_Negocio'].isin(values)]
```

In [12]:
```
df['Ind_Raza_Blanca'] = np.select(
    [
        df['Raza_Desc']=='BLANCO',
        df['Raza_Desc'].isna()
    ],
    [
        'S'
        ,None
    ],
    default='N'
)
```

In [13]:
```
df['Ind_Menarca_Precoz_12'] = np.select(
    [
        df['Num_Edad_Menarca']<12
    ],
    [
        'S'
    ],
    default='N'
)
```

In [14]:
```
df['Ind_Menopausia'] = df.apply(lambda row: 'S' if row['Edad'] >= 47 or row['Ind_Ooforectomia_Bilateral'] == 'Si' or row['Nu
```

In [15]:
```
#from sklearn.cluster import KMeans
#kmeans = KMeans(n_clusters=3, random_state=0).fit(df[['Edad']])
```

In [16]:
```
def clasificar_imc(valor):
    if pd.isna(valor):
        return 'N'
    elif valor < 18.5:
        return 'Bajo_Peso'
    elif 25 <= valor < 30:
        return 'Sobrepeso'
    elif valor >= 30:
        return 'Obesidad'
    else:
        return 'N'

# Crear columnas de clasificación con prefijo 'Ind'
df['Ind_Bajo_Peso'] = df['Valor_IMC'].apply(lambda x: 'S' if clasificar_imc(x) == 'Bajo_Peso' else 'N')
df['Ind_Sobrepeso'] = df['Valor_IMC'].apply(lambda x: 'S' if clasificar_imc(x) in ['Obesidad', 'Sobrepeso'] else 'N')
df['Ind_Obesidad'] = df['Valor_IMC'].apply(lambda x: 'S' if clasificar_imc(x)  == 'Obesidad' else 'N')
```

In [17]:
```

df['Edad_Log'] = np.log(df['Edad'])

df['Edad_Sqrt'] = np.sqrt(df['Edad'])
```

In [18]:
```
df['Ind_Birads_3']= np.select(
    [
        df['Num_Birads'].isin(['3','4','5','6'])
    ],
    [
        'S'
    ],
    default='N'
)
```

In [19]:
```
df['Ind_Obesidad_Posmenopausia'] = np.select(
    [
        (df['Ind_Obesidad']=='S') &
        (df['Ind_Menopausia']=='S')
    ],
    [
        'S'
    ],
    default='N'
)
```

In [20]:
```
#df['Ind_Cigarrillo_Premenopausia'] = np.select(
#    [
#        (df['Ind_Frecuencia_Cigarrillo']=='Si') &
#        (df['Ind_Menopausia']=='N')
#    ],
#    [
#        'S'
#    ],
#    default='N'
#)
```

In [21]:
```
#df['Ind_Cigarrillo_Posmenopausia'] = np.select(
#    [
#        (df['Ind_Frecuencia_Cigarrillo']=='Si') &
#        (df['Ind_Menopausia']=='S')
#    ],
#    [
#        'S'
#    ],
#    default='N'
#)
```

In [22]:
```
df['Ind_Obesidad_Posmenopausia'].value_counts()
```

Out[22]:
```
N    1381894
S     169216
Name: Ind_Obesidad_Posmenopausia, dtype: int64
```

In [23]:
```
#df['Ind_Cigarrillo_Premenopausia'].value_counts()
```

In [24]:
```
#df['Ind_Cigarrillo_Posmenopausia'].value_counts()
```

In [25]:
```
#df = df[df['Num_Birads'].isin(['0','1','2','3','4','5','6'])|df['Num_Birads'].isnull()]
#df = df[df['Num_Birads'].isin(['0','1','2','3'])|df['Num_Birads'].isnull()]
df[df['Num_Birads'].isin(['0','1','2','3'])]
```

Out[25]:

	Afiliado_Id	Asegurado_Id	Dni_Cliente	Fecha_Corte	Dias_EPS	Dias_SEG	Ind_Mayoria_Licor	Ind_Frecuencia_Licor	Sexo_Cd	Raza_Desc
4	1301025.0	NaN	C42881227	2024-01-01	566.0	0.0	No	No	F	MESTIZO
17	920797.0	NaN	C43033822	2024-01-01	575.0	0.0	No	No	F	MESTIZO
19	6619058.0	NaN	C39209855	2024-01-01	460.0	0.0	No	No	F	SIN INFORMACION DESDE LA FUENTE
23	2834682.0	3342995.0	C52587124	2024-01-01	561.0	151.0	No	No	F	SIN INFORMACION DESDE LA FUENTE
25	7499439.0	38000758.0	C1015440326	2024-01-01	499.0	46.0	None	None	F	SIN INFORMACION DESDE LA FUENTE
...	...	...	...	...	...	...	...	...	...	...
2190255	10549074.0	NaN	C1102839072	2024-01-01	462.0	0.0	None	None	F	MESTIZO
2190260	5692225.0	NaN	C66735620	2024-01-01	483.0	0.0	No	No	F	MESTIZO
2190267	827260.0	4427521.0	C39381747	2024-01-01	575.0	700.0	No	No	F	MESTIZO
2190276	1274761.0	NaN	C24487631	2024-01-01	553.0	0.0	No	No	F	MESTIZO
2190278	1423928.0	NaN	C53079615	2024-01-01	563.0	0.0	None	None	F	SIN INFORMACION DESDE LA FUENTE

422770 rows × 37 columns

In [26]:
```
#df['Ind_Birads'] = np.select(
#    [
#        (df['Ind_Birads4']=='S') | (df['Ind_Birads4A']=='S') | (df['Ind_Birads4B']=='S') | (df['Ind_Birads4C']=='S')| (df[
#    ],
#    [
#        'S'
#    ],
#    default='N'
#)
```

In [27]:
```
pd.set_option('display.max_rows', 100)
```

4. Variables y bases de datos

4.1. Presentación de los resultados

Define los parámetros necesarios para crear una directriz clara en la presentación de los distintos resultados

In [28]:
```
# Parámetro que registra si el problema de clasificación es dicotómico o no
bivariate_anal = True
# Parámetro para realizar análisis interactivos (acordeones y grillas), o estáticos
interact_anal = True
```

In [29]:
```
# Elimina el desplazamiento en los resultados de las celdas
remove_scroll_output()
```

4.2. Variables de definición

Define las variables necesarias para identificar y depurar el objeto principal sobre el cual se ejecutará la mayoría de análisis, incluyendo la definición de la variable dependiente.

In [30]:
```
# Nombre que demarca la ejecución del ejercicio, el cual será utilizado para referenciar los resultados
main_name = 'cam'
# Variable de diferenciación de los análisis
dif_var_name = 'Ind_Negocio'
# Nombre de la variable dependiente
dep_var_name = 'Ind_CAM'
# Categoría de éxito de la variable dependiente
    # Por defecto, se considera la categoría de frecuencia menor
dep_var_succ_cat = df[dep_var_name].value_counts().idxmin()
```

4.3. Variables omitidas

4.3.1. Parámetros

Define las variables que no entrarán dentro del conjunto de posibles predictoras para los modelos.

In [31]:
```
omit_vars_names = ['Afiliado_Id', 'Asegurado_Id','Dni_Cliente'#,'Num_Edad_Menopausia'
                    #, 'Ind_Menopausia','Ind_Ooforectomia_Bilateral','Ind_Menopausia_Tardia'
                   ,'Dias_EPS', 'Dias_SEG',
                    'Ind_Mayoria_Licor',
                   'Sexo_Cd',
                   'Fecha_Nacimiento','Fecha_Corte'
                  ]
```

4.3.2. Ejecución

Excluye de la base de datos principal las variables definidas previamente.

In [32]:
```
not_omit_df = non_excluded_data(data=pd.concat([
    df[df['Ind_CAM'] == 'No'].sample(frac=0.2, random_state=42),
    df[df['Ind_CAM'] == 'Si']
])
                                , excl_vars=omit_vars_names)
```

In [33]:
```
not_omit_df.columns
```

Out[33]:
```
Index(['Ind_Frecuencia_Licor', 'Raza_Desc', 'Valor_IMC', 'Num_Edad_Menopausia',
       'Num_Edad_Menarca', 'Ind_Terapia_Hormonal', 'Num_Birads',
       'Ind_Ooforectomia_Bilateral', 'Num_Fam_Primer_Grado_Otros',
       'Num_Fam_Segundo_Grado_Otros', 'Ind_Ant_Fam_Otros',
       'Num_Fam_Primer_Grado_CAM', 'Num_Fam_Segundo_Grado_CAM',
       'Ind_Ant_Fam_CAM', 'Ind_Ant_Radio_Torax', 'Edad', 'Ind_CAM',
       'Ind_Negocio', 'Ind_Raza_Blanca', 'Ind_Menarca_Precoz_12',
       'Ind_Menopausia', 'Ind_Bajo_Peso', 'Ind_Sobrepeso', 'Ind_Obesidad',
       'Edad_Log', 'Edad_Sqrt', 'Ind_Birads_3', 'Ind_Obesidad_Posmenopausia'],
      dtype='object')
```

---

5. Depuración de variables

5.1. Variables categóricas

5.1.1. Depuraciones iniciales

Se realizan todas aquellas depuraciones o transformaciones de las variables categóricas, que se requieren para obtener variables acordes a los modelos a implementar. Este paso puede estar basado, a partir del análisis descriptivo condicional e incondicional de las implicadas. Algunos ejemplos de este tipo de depuraciones, son:

- Creación de nuevas variables
- Cambiar el nombre de algunas variables
- Eliminar algunas categorías
- Imputar algunos registros vacíos

In [34]:
```
# Creación de nuevas variables


# Cambio en el nombre de algunas variables


# Eliminar algunas categorías


# Imputación de registros vacíos


# Otros
```

5.1.2. Categorías por defecto

- **Definición**

A partir de un código prediseñado, se definen las categorías por defecto *(las de mayor frecuencia)* de cada variable categórica.

- Dichas categorías son las que se omitirán en cualquier modelo que lo requiera para no tener multicolinealidad perfecta.

In [35]:
```
default_cats = init_default_categories(data=not_omit_df, dep_var_name=dep_var_name)
default_cats
```

Out[35]:
```
{'Ind_Frecuencia_Licor': 'No',
 'Raza_Desc': 'SIN INFORMACION DESDE LA FUENTE',
 'Ind_Terapia_Hormonal': 'No',
 'Num_Birads': '2',
 'Ind_Ooforectomia_Bilateral': 'No',
 'Ind_Ant_Fam_Otros': 'No',
 'Ind_Ant_Fam_CAM': 'No',
 'Ind_Ant_Radio_Torax': 'No',
 'Ind_Negocio': 'Obligatorios',
 'Ind_Raza_Blanca': 'N',
 'Ind_Menarca_Precoz_12': 'N',
 'Ind_Menopausia': 'N',
 'Ind_Bajo_Peso': 'N',
 'Ind_Sobrepeso': 'S',
 'Ind_Obesidad': 'N',
 'Ind_Birads_3': 'N',
 'Ind_Obesidad_Posmenopausia': 'N'}
```

- **Modificación**

Se corrigen las categorías por defecto, generadas en el paso anterior.

In [36]:
```
#default_cats['Ind_Prolactina_Alta'] = 'N'
#default_cats['Ind_Sobrepeso'] = 'N'
#default_cats['Ind_Ant_Fam_CA'] = 'N'
#default_cats['Ind_Licor'] = 'N'
#Antecedente_Familia', 'Ind_Embarazo', 'Numero_Embarazos_Cat', 'Numero_Embarazos_Cat_Red

#default_cats['Antecedente_Familia'] = 'No'
#default_cats['Ind_Embarazo'] = 'No'
#default_cats['Numero_Embarazos_Cat'] = str(0)
#default_cats['Numero_Embarazos_Cat_Red'] = str(0)
default_cats['Ind_Sobrepeso'] = 'N'
```

5.2. Variables Numéricas

5.2.1. Depuraciones iniciales

Se realizan todas aquellas depuraciones o transformaciones de las variables numéricas, que se requieren para obtener variables acordes a los modelos a implementar. Este paso puede estar basado, a partir del análisis descriptivo condicional e incondicional de las implicadas. Algunos ejemplos de este tipo de depuraciones, son:

> - Cambiar el nombre de algunas variables
> - Eliminar algunas observaciones
> - Realizar transformaciones sobre los datos
> - Imputar algunos registros vacíos

In [37]:
```
# Cambio en el nombre de algunas variables


# Eliminar algunas observaciones


# Transformaciones sobre los datos


# Imputación de registros vacíos


# Otros
```

Tabla de Contenido

---

1. Datos Faltantes

1.1. Análisis Incondicional

Tablas y gráficos que dan cuenta de la estructura gerenal de completitud de las variables, y de la relación de observaciones faltantes entre las mismas.

In [38]:
```
print_null_count_analysis(data=not_omit_df , dif_var_name=dif_var_name, interact=interact_anal)
```

▾ **Estadísticos Resumen**

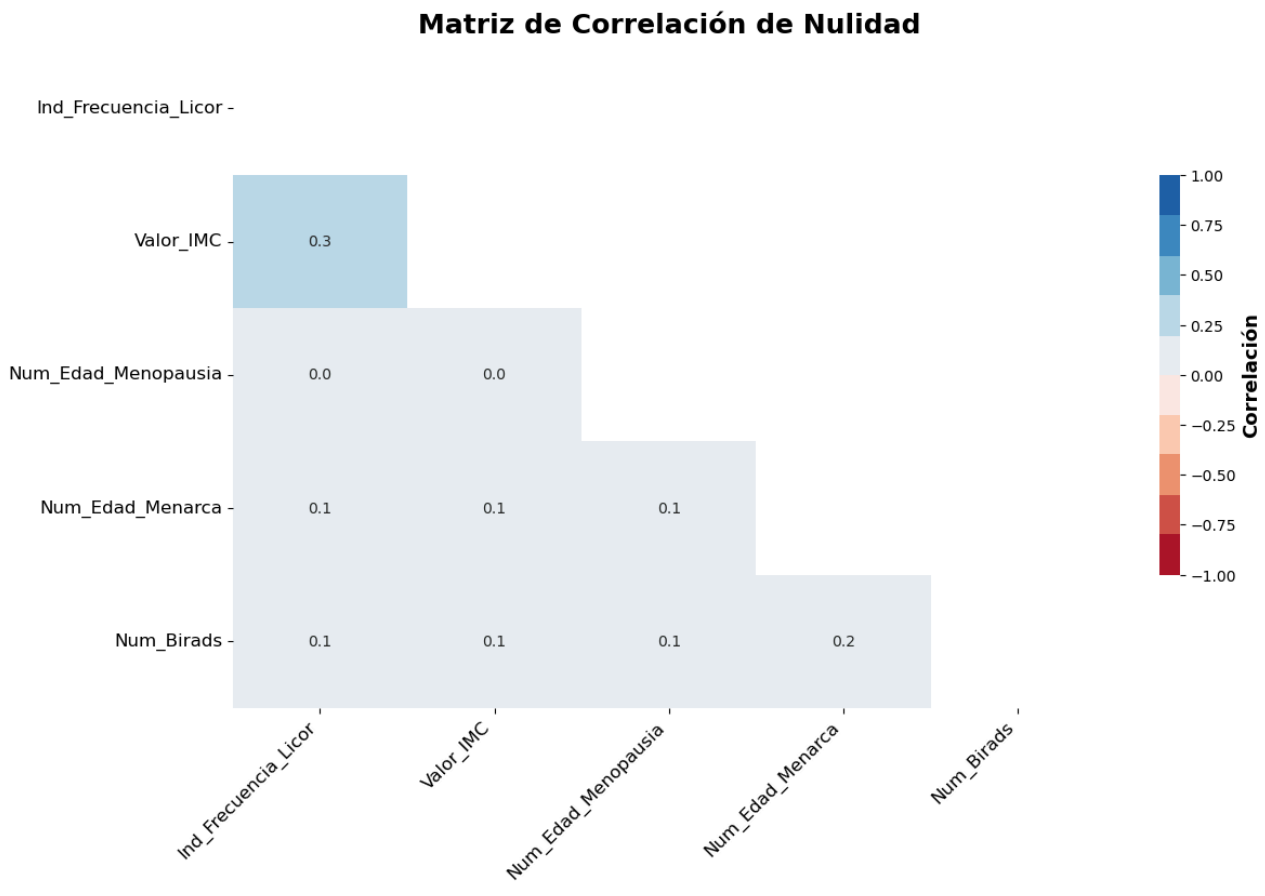
Frecuencia | Frec. Condic.

Detalle de Registros Nulos

Variable	Reg. vacíos	Reg. llenos	% Reg. vacíos	% Reg. llenos
Num_Edad_Menopausia	318,659	4,457	98.62%	1.38%
Num_Birads	230,978	92,138	71.48%	28.52%
Num_Edad_Menarca	167,790	155,326	51.93%	48.07%
Ind_Frecuencia_Licor	108,458	214,658	33.57%	66.43%
Valor_IMC	34,904	288,212	10.80%	89.20%

▾ **Gráficos Descriptivos**

Correlación | Agrupamiento



1.2. Exclusión de variables (n°1)

Define la base de datos, que además de no considerar variables omitidas desde el inicio, también excluye aquellas que por la cantidad de registros vacíos, no harían parte de la modelación.

In [39]:
```
df.columns
```

Out[39]:
```
Index(['Afiliado_Id', 'Asegurado_Id', 'Dni_Cliente', 'Fecha_Corte', 'Dias_EPS',
       'Dias_SEG', 'Ind_Mayoria_Licor', 'Ind_Frecuencia_Licor', 'Sexo_Cd',
       'Raza_Desc', 'Valor_IMC', 'Num_Edad_Menopausia', 'Num_Edad_Menarca',
       'Ind_Terapia_Hormonal', 'Num_Birads', 'Ind_Ooforectomia_Bilateral',
       'Num_Fam_Primer_Grado_Otros', 'Num_Fam_Segundo_Grado_Otros',
       'Ind_Ant_Fam_Otros', 'Num_Fam_Primer_Grado_CAM',
       'Num_Fam_Segundo_Grado_CAM', 'Ind_Ant_Fam_CAM', 'Ind_Ant_Radio_Torax',
       'Fecha_Nacimiento', 'Edad', 'Ind_CAM', 'Ind_Negocio', 'Ind_Raza_Blanca',
       'Ind_Menarca_Precoz_12', 'Ind_Menopausia', 'Ind_Bajo_Peso',
       'Ind_Sobrepeso', 'Ind_Obesidad', 'Edad_Log', 'Edad_Sqrt',
       'Ind_Birads_3', 'Ind_Obesidad_Posmenopausia'],
      dtype='object')
```

1.2.1. Parámetros

In [40]:
```
# Nombre de las variables a excluir por cantidad de registros, de tipo: 'list'



#excl_vars_names_1 = []

excl_vars_names_1 = ['Ind_Ooforectomia_Bilateral','Num_Edad_Menopausia','Num_Birads'
                     ]
```

1.2.2. Ejecución

In [41]:
```
not_excl_df = non_excluded_data(data=not_omit_df, excl_vars=excl_vars_names_1)
```

In [42]:
```
not_excl_df.columns
```

Out[42]:
```
Index(['Ind_Frecuencia_Licor', 'Raza_Desc', 'Valor_IMC', 'Num_Edad_Menarca',
       'Ind_Terapia_Hormonal', 'Num_Fam_Primer_Grado_Otros',
       'Num_Fam_Segundo_Grado_Otros', 'Ind_Ant_Fam_Otros',
       'Num_Fam_Primer_Grado_CAM', 'Num_Fam_Segundo_Grado_CAM',
       'Ind_Ant_Fam_CAM', 'Ind_Ant_Radio_Torax', 'Edad', 'Ind_CAM',
       'Ind_Negocio', 'Ind_Raza_Blanca', 'Ind_Menarca_Precoz_12',
       'Ind_Menopausia', 'Ind_Bajo_Peso', 'Ind_Sobrepeso', 'Ind_Obesidad',
       'Edad_Log', 'Edad_Sqrt', 'Ind_Birads_3', 'Ind_Obesidad_Posmenopausia'],
      dtype='object')
```

In [43]:
```
#not_excl_df=not_excl_df[df['Ind_Menopausia']=='S']
```

1.3. Análisis Condicional

Tablas y gráficos que dan cuenta de la estructura de completitud de las variables, en torno a los distintos grupos que forman las categorías de la variable dependiente.

- En esta sección se evalúa la opción de considerar el conjunto de datos con registros completos, como la base final para realizar los modelos. El elemento clave de este análisis, es el cambio de escenario de desbalanceo de la variable dependiente.

In [44]:
```
st = time.time()
print_cond_null_analysis(data=not_excl_df,
                         dep_var_name=dep_var_name,
                         dif_var_name=dif_var_name,
                         interact=interact_anal)
```

▾ **Estadísticos y Pruebas**

Frecuencia | Independencia | Frec. Condic.

Conteo de registros

	Frecuencias			Proporciones	
Ind_CAM	**No**	**Si**	**Total**	**No**	**Si**
Registros					
Con Dato	101,199	4,890	106,089	95.39%	4.61%
Sin Dato	205,799	11,228	217,027	94.83%	5.17%
Total	306,998	16,118	323,116	95.01%	4.99%

In [45]:
```
ed = time.time()
print(ed-st)
```

```
2.500866413116455
```

In [46]:
```
not_omit_df[['Ind_Negocio']].count()
```

Out[46]:
```
Ind_Negocio    323116
dtype: int64
```

1.4. Exclusión de variables (n°2)

Excluye las variables que fueron necesarias para los anteriores análisis, pero que no serán consideradas como candidatas explicativas en los modelos.

1.4.1. Parámetros

In [47]:
```
#excl_vars_names_2 = [dif_var_name]
excl_vars_names_2 = [dif_var_name]
```

1.4.2. Ejecución

In [48]:
```
not_excl_df = non_excluded_data(data=not_excl_df, excl_vars=excl_vars_names_2)
```

2. Efectos Individuales

2.1. Definición de parámetros

Define los parámetros para realizar pruebas estadísticas y determinar el análisis bivariado por tipos de variables.

In [49]:
```
# Nivel de significancia para realizar las pruebas estadísticas
alpha_num = 0.05
# Momentos centrales para las variables numéricas
central_moments = ['mean', 'median']
# Pruebas de diferencias de momentos y distribuciones, para las variables numéricas:
    # 'wilctest': Prueba de diferencia de medias de Wilcoxon
    # 'krusktest': Prueba de diferencia de medianas de Kruskal Wallis
    # 'kstest': Pruebas de diferencia de distribuciones de Kolmogorov Smirnov
diff_tests = ['wilctest', 'krusktest', 'kstest']
```

2.2. Desbalanceo por variable

Detalla la magnitud del cambio en el desbalanceo de la población, al considerar solamente las observaciones con registros completos, por cada una de las variables.

In [50]:
```
st = time.time()
print_detailed_imbalance_table(data=not_excl_df,
                               dep_var_name=dep_var_name,
                               interact=interact_anal)
```

Desbalanceo con Reg. Completos

Categóricas | Numéricas

		Frecuencias		Proporciones	
Ind_CAM	No	Si	Total	No	Si
Variables					
Ind_Frecuencia_Licor	206,987	7,671	214,658	96.43%	3.57%
Raza_Desc	306,998	16,118	323,116	95.01%	4.99%
Ind_Terapia_Hormonal	306,998	16,118	323,116	95.01%	4.99%
Ind_Ant_Fam_Otros	306,998	16,118	323,116	95.01%	4.99%
Ind_Ant_Fam_CAM	306,998	16,118	323,116	95.01%	4.99%
Ind_Ant_Radio_Torax	306,998	16,118	323,116	95.01%	4.99%
Ind_Raza_Blanca	306,998	16,118	323,116	95.01%	4.99%
Ind_Menarca_Precoz_12	306,998	16,118	323,116	95.01%	4.99%
Ind_Menopausia	306,998	16,118	323,116	95.01%	4.99%
Ind_Bajo_Peso	306,998	16,118	323,116	95.01%	4.99%
Ind_Sobrepeso	306,998	16,118	323,116	95.01%	4.99%
Ind_Obesidad	306,998	16,118	323,116	95.01%	4.99%
Ind_Birads_3	306,998	16,118	323,116	95.01%	4.99%
Ind_Obesidad_Posmenopausia	306,998	16,118	323,116	95.01%	4.99%

In [51]:
```
ed = time.time()
print(ed-st)
```

```
13.023013591766357
```

In [52]:
```
columns = not_excl_df.columns
for column in columns:
    print(column, not_excl_df[column].dtype)
```

```
Ind_Frecuencia_Licor object
Raza_Desc object
Valor_IMC float64
Num_Edad_Menarca float64
Ind_Terapia_Hormonal object
Num_Fam_Primer_Grado_Otros float64
Num_Fam_Segundo_Grado_Otros float64
Ind_Ant_Fam_Otros object
Num_Fam_Primer_Grado_CAM float64
Num_Fam_Segundo_Grado_CAM float64
Ind_Ant_Fam_CAM object
Ind_Ant_Radio_Torax object
Edad float64
Ind_CAM object
Ind_Raza_Blanca object
Ind_Menarca_Precoz_12 object
Ind_Menopausia object
Ind_Bajo_Peso object
Ind_Sobrepeso object
Ind_Obesidad object
Edad_Log float64
Edad_Sqrt float64
Ind_Birads_3 object
Ind_Obesidad_Posmenopausia object
```

In [53]:
```
not_excl_df.columns
```

Out[53]:
```
Index(['Ind_Frecuencia_Licor', 'Raza_Desc', 'Valor_IMC', 'Num_Edad_Menarca',
       'Ind_Terapia_Hormonal', 'Num_Fam_Primer_Grado_Otros',
       'Num_Fam_Segundo_Grado_Otros', 'Ind_Ant_Fam_Otros',
       'Num_Fam_Primer_Grado_CAM', 'Num_Fam_Segundo_Grado_CAM',
       'Ind_Ant_Fam_CAM', 'Ind_Ant_Radio_Torax', 'Edad', 'Ind_CAM',
       'Ind_Raza_Blanca', 'Ind_Menarca_Precoz_12', 'Ind_Menopausia',
       'Ind_Bajo_Peso', 'Ind_Sobrepeso', 'Ind_Obesidad', 'Edad_Log',
       'Edad_Sqrt', 'Ind_Birads_3', 'Ind_Obesidad_Posmenopausia'],
      dtype='object')
```

In [54]:
```
#col_list = ['Ind_CAM',
#       'Ind_Frecuencia_Cigarrillo', 'Ind_Frecuencia_Licor',
#       'Ind_Frecuencia_Ejercicio', 'Raza_Desc', 'Valor_IMC',
#       'Num_Edad_Menarca', 'Ind_Terapia_Hormonal',
#       'Num_Fam_Primer_Grado_Otros', 'Num_Fam_Segundo_Grado_Otros',
#       'Ind_Ant_Fam_Otros', 'Num_Fam_Primer_Grado_CAM',
#       'Num_Fam_Segundo_Grado_CAM', 'Ind_Ant_Fam_CAM', 'Edad',
#      'Ind_Raza_Blanca', 'Ind_Menarca_Precoz_12', 'Ind_Bajo_Peso',
#       'Ind_Sobrepeso', 'Ind_Obesidad', 'Edad_Log', 'Edad_Sqrt',
#       'Ind_Birads_3',
#            'Ind_Obesidad_Posmenopausia','Ind_Cigarrillo_Premenopausia',
#            'Ind_Cigarrillo_Posmenopausia',
#            'Ind_Menopausia'
#          ]
```

2.3. Estimación de efectos

Se realiza un análisis de efectos explicativos bivariados *(dependiente y una independiente)*. Para ello, se emplean pruebas de diferencias de momentos, distribuciones y regresiones logísiticas. Con este análisis se pretende observar cuál es el signo y magnitud del efecto de una variable por sí sola en la explicación del fenómeno y cómo cambian estas condiciones, cuando involucro datos completos.

- El análisis se ejecuta, tanto para el conjunto de datos completo por variable, como el obtenido al considerar sólo los registros llenos para todas las variables.

2.3.1. Registros completos bivariados

Evaluación de los efectos individuales sobre la base de datos que se forma al seleccionar registros completos, al escoger solo dos variables al tiempo.

In [55]:
```
st = time.time()
print_bivar_infer_analysis(data=not_excl_df,
                           dep_var_name=dep_var_name,
                           dep_var_succ=dep_var_succ_cat,
                           alpha=alpha_num,
                           cat_ind_vars_def=default_cats,
                           meas=central_moments,
                           test=diff_tests,
                           complt_db=False,
                           interact=interact_anal)
```

▾ **Efectos Explicativos Individuales**

Categóricas | Numéricas

		Proporciones		Independ. (χ^2)	Odds-Ratios (OR): Regresión Logística				
	Ind_CAM	No	Si	Valor P.	Log_Odd	Valor P.	OR	Inf. (5%)	Sup. (95%)
Variable	Categoría								
Ind_Frecuencia_Licor	No	97.39%	2.61%	nan%	nan	nan%	nan	nan	nan
	Si	92.32%	7.68%	0.00%	1.13	0.00%	3.10	2.96	3.25
Raza_Desc	AFROAMERICANO	94.12%	5.88%	2.38%	0.59	0.00%	1.81	1.55	2.10
	BLANCO	89.86%	10.14%	0.00%	1.18	0.00%	3.27	3.11	3.43
	INDÍGENA	92.19%	7.81%	45.28%	0.90	5.42%	2.45	0.98	6.11
	MESTIZO	92.32%	7.68%	0.00%	0.88	0.00%	2.41	2.33	2.49
	MULATO	92.93%	7.07%	8.00%	0.79	0.01%	2.20	1.49	3.26
	SIN INFORMACION DESDE LA FUENTE	96.66%	3.34%	0.00%	nan	nan%	nan	nan	nan
	ZAMBO	91.95%	8.05%	9.31%	0.93	0.09%	2.53	1.47	4.37
Ind_Terapia_Hormonal	No	95.03%	4.97%	nan%	nan	nan%	nan	nan	nan
	Si	89.46%	10.54%	0.00%	0.81	0.00%	2.25	1.79	2.83
Ind_Ant_Fam_Otros	No	95.24%	4.76%	nan%	nan	nan%	nan	nan	nan
	Si	91.38%	8.62%	0.00%	0.63	0.00%	1.89	1.79	1.99
Ind_Ant_Fam_CAM	No	95.19%	4.81%	nan%	nan	nan%	nan	nan	nan
	Si	65.06%	34.94%	0.00%	2.36	0.00%	10.63	9.66	11.69
Ind_Ant_Radio_Torax	No	94.28%	5.72%	nan%	nan	nan%	nan	nan	nan
	Si	98.00%	2.00%	0.00%	-1.09	0.00%	0.34	0.32	0.36
Ind_Raza_Blanca	N	95.43%	4.57%	nan%	nan	nan%	nan	nan	nan
	S	89.86%	10.14%	0.00%	0.86	0.00%	2.36	2.25	2.47
Ind_Menarca_Precoz_12	N	95.82%	4.18%	nan%	nan	nan%	nan	nan	nan
	S	92.62%	7.38%	0.00%	0.60	0.00%	1.83	1.77	1.89
Ind_Menopausia	N	97.07%	2.93%	nan%	nan	nan%	nan	nan	nan
	S	92.88%	7.12%	0.00%	0.93	0.00%	2.54	2.45	2.63
Ind_Bajo_Peso	N	95.02%	4.98%	nan%	nan	nan%	nan	nan	nan
	S	94.62%	5.38%	32.83%	0.08	30.85%	1.08	0.93	1.27
Ind_Sobrepeso	N	94.58%	5.42%	nan%	nan	nan%	nan	nan	nan
	S	95.35%	4.65%	0.00%	-0.16	0.00%	0.85	0.82	0.88
Ind_Obesidad	N	94.88%	5.12%	nan%	nan	nan%	nan	nan	nan
	S	95.51%	4.49%	0.00%	-0.14	0.00%	0.87	0.84	0.91
Ind_Birads_3	N	95.14%	4.86%	nan%	nan	nan%	nan	nan	nan
	S	91.66%	8.34%	0.00%	0.58	0.00%	1.78	1.67	1.91
Ind_Obesidad_Posmenopausia	N	95.21%	4.79%	nan%	nan	nan%	nan	nan	nan
	S	93.41%	6.59%	0.00%	0.34	0.00%	1.40	1.34	1.47

In [56]:
```
ed = time.time()
print(ed-st)
```

```
24.44866108894348
```

In [57]:
```
#imc_mean = not_excl_df['IMC'].mean()
#menarc_mean = not_excl_df['Num_Edad_Menarca'].median()
#menop_mean = not_excl_df['Num_Edad_Menopausia'].median()
```

In [58]:
```
#not_excl_df['IMC']=not_excl_df['IMC'].fillna(imc_mean)
#not_excl_df['Num_Edad_Menarca']=not_excl_df['Num_Edad_Menarca'].fillna(menarc_mean)
#not_excl_df['Num_Edad_Menopausia']=not_excl_df['Num_Edad_Menopausia'].fillna(menop_mean)
```

In [59]:
```
#not_excl_df.head()
```

In [60]:
```
#not_excl_df['Ind_Embarazo'].value_counts()
```

In [61]:
```
#not_excl_df[not_excl_df['Ind_Embarazo']=='Si'& not_excl_df['Ind_Embarazo']=='Si']
#not_excl_df.loc[(not_excl_df['Ind_Embarazo'] =='No') & (not_excl_df['Ind_Cancer_Mama'] =='Si')]
```

In [62]:
```
#not_excl_df.count()
```

2.3.2. Registros completos multivariados

Evaluación de los efectos individuales sobre la base de datos que se forma al seleccionar registros completos, al escoger todas las variables al tiempo.

In [63]:
```
not_excl_df.columns
```

Out[63]:
```
Index(['Ind_Frecuencia_Licor', 'Raza_Desc', 'Valor_IMC', 'Num_Edad_Menarca',
       'Ind_Terapia_Hormonal', 'Num_Fam_Primer_Grado_Otros',
       'Num_Fam_Segundo_Grado_Otros', 'Ind_Ant_Fam_Otros',
       'Num_Fam_Primer_Grado_CAM', 'Num_Fam_Segundo_Grado_CAM',
       'Ind_Ant_Fam_CAM', 'Ind_Ant_Radio_Torax', 'Edad', 'Ind_CAM',
       'Ind_Raza_Blanca', 'Ind_Menarca_Precoz_12', 'Ind_Menopausia',
       'Ind_Bajo_Peso', 'Ind_Sobrepeso', 'Ind_Obesidad', 'Edad_Log',
       'Edad_Sqrt', 'Ind_Birads_3', 'Ind_Obesidad_Posmenopausia'],
      dtype='object')
```

In [64]:
```
st = time.time()
print_bivar_infer_analysis(data=not_excl_df,
                           dep_var_name=dep_var_name,
                           dep_var_succ=dep_var_succ_cat,
                           alpha=alpha_num,
                           cat_ind_vars_def=default_cats,
                           meas=central_moments,
                           test=diff_tests,
                           complt_db=False,
                           interact=interact_anal
                          )
```

▾ **Efectos Explicativos Individuales**

Categóricas | Numéricas

		Proporciones		Independ. (χ^2)	Odds-Ratios (OR): Regresión Logística				
	Ind_CAM	No	Si	Valor P.	Log_Odd	Valor P.	OR	Inf. (5%)	Sup. (95%)
Variable	Categoría								
Ind_Frecuencia_Licor	No	97.39%	2.61%	nan%	nan	nan%	nan	nan	nan
	Si	92.32%	7.68%	0.00%	1.13	0.00%	3.10	2.96	3.25
Raza_Desc	AFROAMERICANO	94.12%	5.88%	2.38%	0.59	0.00%	1.81	1.55	2.10
	BLANCO	89.86%	10.14%	0.00%	1.18	0.00%	3.27	3.11	3.43
	INDÍGENA	92.19%	7.81%	45.28%	0.90	5.42%	2.45	0.98	6.11
	MESTIZO	92.32%	7.68%	0.00%	0.88	0.00%	2.41	2.33	2.49
	MULATO	92.93%	7.07%	8.00%	0.79	0.01%	2.20	1.49	3.26
	SIN INFORMACION DESDE LA FUENTE	96.66%	3.34%	0.00%	nan	nan%	nan	nan	nan
	ZAMBO	91.95%	8.05%	9.31%	0.93	0.09%	2.53	1.47	4.37
Ind_Terapia_Hormonal	No	95.03%	4.97%	nan%	nan	nan%	nan	nan	nan
	Si	89.46%	10.54%	0.00%	0.81	0.00%	2.25	1.79	2.83
Ind_Ant_Fam_Otros	No	95.24%	4.76%	nan%	nan	nan%	nan	nan	nan
	Si	91.38%	8.62%	0.00%	0.63	0.00%	1.89	1.79	1.99
Ind_Ant_Fam_CAM	No	95.19%	4.81%	nan%	nan	nan%	nan	nan	nan
	Si	65.06%	34.94%	0.00%	2.36	0.00%	10.63	9.66	11.69
Ind_Ant_Radio_Torax	No	94.28%	5.72%	nan%	nan	nan%	nan	nan	nan
	Si	98.00%	2.00%	0.00%	-1.09	0.00%	0.34	0.32	0.36
Ind_Raza_Blanca	N	95.43%	4.57%	nan%	nan	nan%	nan	nan	nan
	S	89.86%	10.14%	0.00%	0.86	0.00%	2.36	2.25	2.47
Ind_Menarca_Precoz_12	N	95.82%	4.18%	nan%	nan	nan%	nan	nan	nan
	S	92.62%	7.38%	0.00%	0.60	0.00%	1.83	1.77	1.89
Ind_Menopausia	N	97.07%	2.93%	nan%	nan	nan%	nan	nan	nan
	S	92.88%	7.12%	0.00%	0.93	0.00%	2.54	2.45	2.63
Ind_Bajo_Peso	N	95.02%	4.98%	nan%	nan	nan%	nan	nan	nan
	S	94.62%	5.38%	32.83%	0.08	30.85%	1.08	0.93	1.27
Ind_Sobrepeso	N	94.58%	5.42%	nan%	nan	nan%	nan	nan	nan
	S	95.35%	4.65%	0.00%	-0.16	0.00%	0.85	0.82	0.88
Ind_Obesidad	N	94.88%	5.12%	nan%	nan	nan%	nan	nan	nan
	S	95.51%	4.49%	0.00%	-0.14	0.00%	0.87	0.84	0.91
Ind_Birads_3	N	95.14%	4.86%	nan%	nan	nan%	nan	nan	nan
	S	91.66%	8.34%	0.00%	0.58	0.00%	1.78	1.67	1.91
Ind_Obesidad_Posmenopausia	N	95.21%	4.79%	nan%	nan	nan%	nan	nan	nan
	S	93.41%	6.59%	0.00%	0.34	0.00%	1.40	1.34	1.47

In [65]:
```
ed= time.time()
print(ed-st)
```

```
25.855847597122192
```

2.4. Exclusión de variables (n°3)

Excluye las variables que no tienen el signo y significancia esperada *(desde la teoría)*, a partir de los análisis bivariados, y cuyo resultado no provendría de la presencia de variables confusoras. Así mismo, se eliminan aquellas variables cuyo efecto cambia sustancialmente cuando se considera la base de datos con registros completos para todas las variables.

2.4.1. Parámetros

In [66]:
```
#excl_vars_names_3 = ['Ind_Frecuencia_Cigarrillo',
#                      'Ind_Frecuencia_Licor',
#                      'Raza_Desc',
#                      'Dias_HTA',
#                      'Valor_PAD']
excl_vars_names_3 = ['Raza_Desc'
                     ,'Num_Edad_Menarca','Num_Fam_Primer_Grado_Otros','Num_Fam_Segundo_Grado_Otros',
                     'Num_Fam_Primer_Grado_CAM', 'Num_Fam_Segundo_Grado_CAM', 'Ind_Ant_Radio_Torax'

       ]
```

In [67]:
```
not_excl_df.columns
```

Out[67]:
```
Index(['Ind_Frecuencia_Licor', 'Raza_Desc', 'Valor_IMC', 'Num_Edad_Menarca',
       'Ind_Terapia_Hormonal', 'Num_Fam_Primer_Grado_Otros',
       'Num_Fam_Segundo_Grado_Otros', 'Ind_Ant_Fam_Otros',
       'Num_Fam_Primer_Grado_CAM', 'Num_Fam_Segundo_Grado_CAM',
       'Ind_Ant_Fam_CAM', 'Ind_Ant_Radio_Torax', 'Edad', 'Ind_CAM',
       'Ind_Raza_Blanca', 'Ind_Menarca_Precoz_12', 'Ind_Menopausia',
       'Ind_Bajo_Peso', 'Ind_Sobrepeso', 'Ind_Obesidad', 'Edad_Log',
       'Edad_Sqrt', 'Ind_Birads_3', 'Ind_Obesidad_Posmenopausia'],
      dtype='object')
```

2.4.2. Ejecución

In [68]:
```
pre_model_data = non_excluded_data(data=not_excl_df.dropna(), excl_vars=excl_vars_names_3)
```

In [69]:
```
pre_model_data.columns
```

Out[69]:
```
Index(['Ind_Frecuencia_Licor', 'Valor_IMC', 'Ind_Terapia_Hormonal',
       'Ind_Ant_Fam_Otros', 'Ind_Ant_Fam_CAM', 'Edad', 'Ind_CAM',
       'Ind_Raza_Blanca', 'Ind_Menarca_Precoz_12', 'Ind_Menopausia',
       'Ind_Bajo_Peso', 'Ind_Sobrepeso', 'Ind_Obesidad', 'Edad_Log',
       'Edad_Sqrt', 'Ind_Birads_3', 'Ind_Obesidad_Posmenopausia'],
      dtype='object')
```

In [70]:
```
pre_model_data.count()[0]
```

Out[70]:
```
106089
```

---

3. Estructura de dependencia

Define una serie de análisis y gráficos de correlación, a fin de entender el nivel de relación *(no sólo lineal)*, que tienen las variables explicativas entre sí, como acercamiento al concepto de **Multicolinealidad** *(en modelos de regresión)*.

3.1. Numéricas

3.1.1. Definición de parámetros

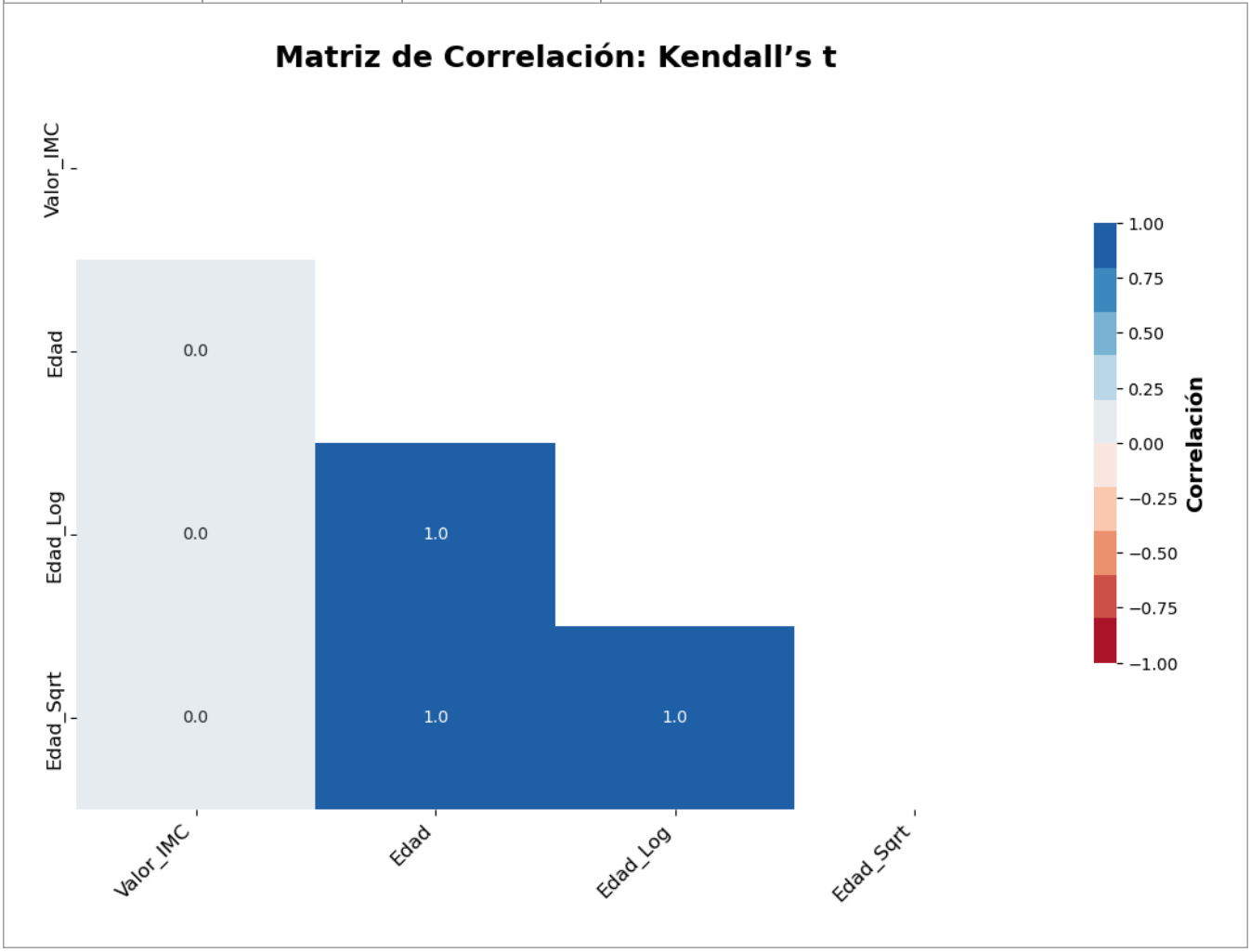
In [71]:
```
# Tipo de variable determinante del análisis
    # 'num': Variables numéricas
    # 'cat': Variables categóricas
num_corr_analysis_type = 'num'
# Nombres de los métodos de análisis de correlación, disponibles por tipo de variable:
    # 'kendall': Correlación Kendall's Tau - > Variables numéricas
    # 'pearson': Correlación de Pearson - > Variables numéricas
    # 'spearman': Correlación de Spearman - > Variables numéricas
    # 'cramer': Correlación corregida de Cramer - > Variables categóricas
num_corr_method_names = ['kendall', 'pearson', 'spearman']
# Umbral de decisión para analizar posible colinealidad, en base a coeficientes de correlación
num_corr_threshold_val = 0.5
```

3.1.2. Ejecución

In [72]:
```
print_multicol_analysis(data=pre_model_data,
                        dep_var_name=dep_var_name,
                        anal_type=num_corr_analysis_type,
                        method=num_corr_method_names,
                        threshold=num_corr_threshold_val,
                        interact=interact_anal)
```

Gráficos Descriptivos

Kendall's t | Pearson's r | Spearman's rho



▾ **Resumen de Estadísticos**

Correlación

Resumen de correlaciones

Método	Variable Fila	Variable Columna
Kendall's t	**Edad**	2
	Edad_Log	2
	Edad_Sqrt	2
Pearson's r	**Edad**	2
	Edad_Log	2
	Edad_Sqrt	2
Spearman's rho	**Edad**	2
	Edad_Log	2
	Edad_Sqrt	2

Detalle de correlaciones

Método	Variable Fila	Variable Columna	Corr.
Kendall's t	Edad	Edad_Log	1.0
Kendall's t	Edad	Edad_Sqrt	1.0
Kendall's t	Edad_Log	Edad	1.0
Kendall's t	Edad_Log	Edad_Sqrt	1.0
Kendall's t	Edad_Sqrt	Edad	1.0
Kendall's t	Edad_Sqrt	Edad_Log	1.0
Pearson's r	Edad	Edad_Log	1.0
Pearson's r	Edad	Edad_Sqrt	1.0
Pearson's r	Edad_Log	Edad	1.0
Pearson's r	Edad_Log	Edad_Sqrt	1.0
Pearson's r	Edad_Sqrt	Edad	1.0
Pearson's r	Edad_Sqrt	Edad_Log	1.0
Spearman's rho	Edad	Edad_Log	1.0
Spearman's rho	Edad	Edad_Sqrt	1.0
Spearman's rho	Edad_Log	Edad	1.0
Spearman's rho	Edad_Log	Edad_Sqrt	1.0
Spearman's rho	Edad_Sqrt	Edad	1.0
Spearman's rho	Edad_Sqrt	Edad_Log	1.0

3.2. Categóricas

3.2.1. Definición de parámetros

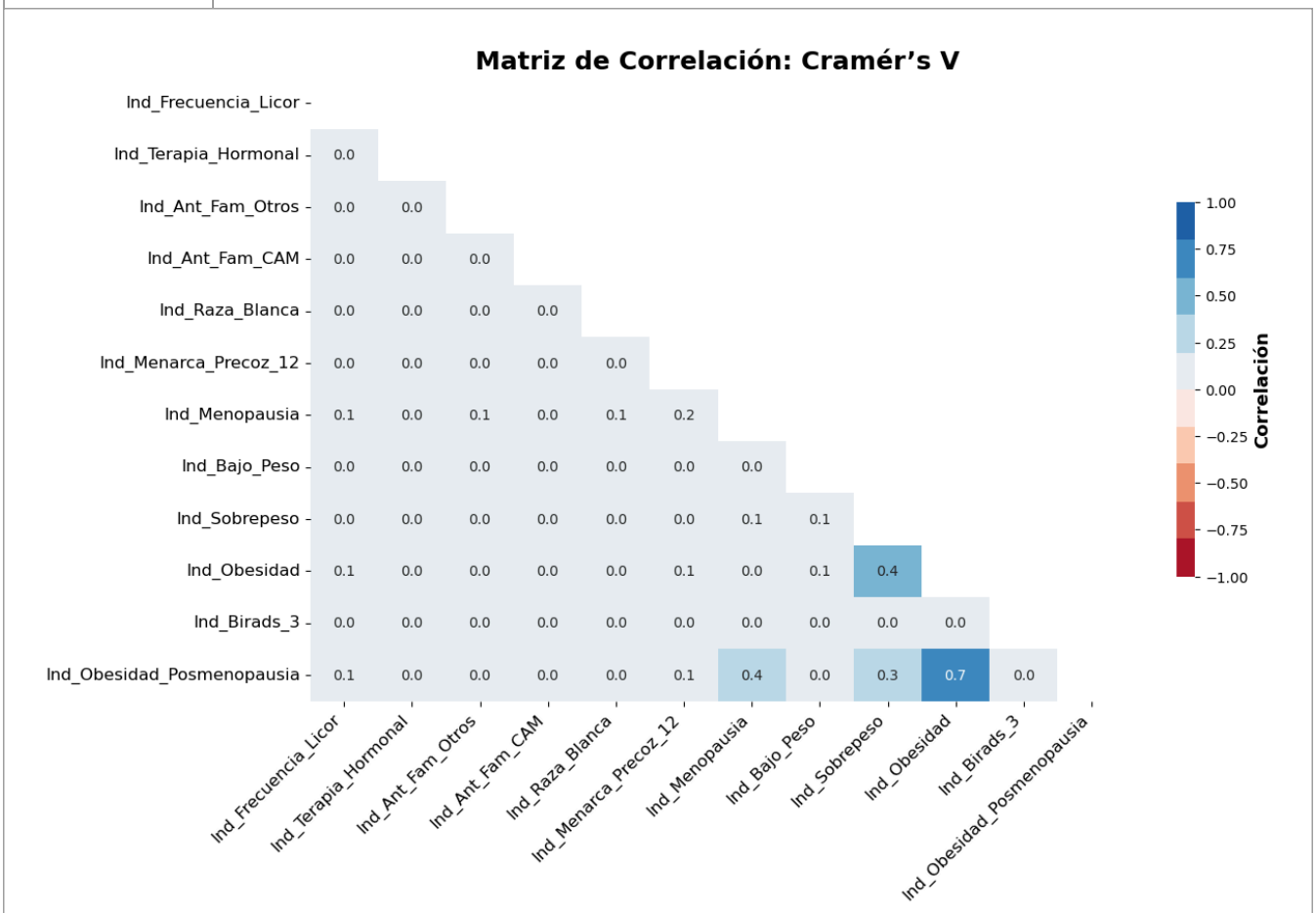
In [73]:
```
# Tipo de variable determinante del análisis
    # 'num': Variables numéricas
    # 'cat': Variables categóricas
cat_corr_analysis_type = 'cat'
# Nombres de los métodos de análisis de correlación, disponibles por tipo de variable:
    # 'kendall': Correlación Kendall's Tau - > Variables numéricas
    # 'pearson': Correlación de Pearson - > Variables numéricas
    # 'spearman': Correlación de Spearman - > Variables numéricas
    # 'cramer': Correlación corregida de Cramer - > Variables categóricas
cat_corr_method_names = ['cramer']
# Umbral de decisión para analizar posible colinealidad, en base a coeficientes de correlación
cat_corr_threshold_val = 0.6
```

3.2.2. Ejecución

In [74]:
```
print_multicol_analysis(data=pre_model_data,
                        dep_var_name=dep_var_name,
                        anal_type=cat_corr_analysis_type,
                        method=cat_corr_method_names,
                        threshold=cat_corr_threshold_val,
                        interact=interact_anal)
```

▾ **Gráficos Descriptivos**

Cramér's V



▾ **Resumen de Estadísticos**

Correlación

Resumen de correlaciones

Método	Variable Fila	Variable Columna
Cramér's V	**Ind_Obesidad**	1
	Ind_Obesidad_Posmenopausia	1

Detalle de correlaciones

Método	Variable Fila	Variable Columna	Corr.
Cramér's V	Ind_Obesidad	Ind_Obesidad_Posmenopausia	0.7
Cramér's V	Ind_Obesidad_Posmenopausia	Ind_Obesidad	0.7

4. Elección entre variables relacionadas

En aras de evitar la presencia al tiempo de variables redundantes (desde un punto de vista de dependencia lineal), se elige una variable por grupo de candidatas. La elección se lleva a cabo a partir de un análisis de importancia o mejora en la bondad de ajuste del modelo.

4.1. Definición de parámetros

Definición de los conjuntos de variables redundantes que harían parte del análisis.

In [75]:
```
# Lista de listas, con cada grupo de variables a evaluar
corr_var_groups_1 = [['Edad','Edad_Log','Edad_Sqrt']]
```

4.2. Ejecución

In [76]:
```
print_corr_vars_selection(data=pre_model_data,
                          dep_var_name=dep_var_name,
                          dep_var_succ=dep_var_succ_cat,
                          corr_vars=corr_var_groups_1,
                          cat_ind_vars_def=default_cats,
                          bivar_anal=bivariate_anal,
                          interact=interact_anal)
```

▾ **Elección de Variables Colineales**

Ranking | Selección

Ordenamiento por importancia

Grupo	Variables	Ranking
1	Edad_Log	1
1	Edad_Sqrt	2
1	Edad	3

In [77]:
```
#print_corr_vars_selection(data=pre_model_data,
#                          dep_var_name=dep_var_name,
#                          dep_var_succ=dep_var_succ_cat,
#                          corr_vars=corr_var_groups_2,
#                          cat_ind_vars_def=default_cats,
#                          bivar_anal=bivariate_anal,
#                          interact=interact_anal)
```

4.3. Exclusión de variables (n°4)

Define la base de datos que, además de no considerar variables omitidas, excluidas por cantidad de registros vacíos y calidad de datos, también excluye aquellas que podrían ser redundantes. La salida de este proceso, conforma la base de datos de pre-modelación.

4.3.1. Parámetros

In [78]:
```
pre_model_data.columns
```

Out[78]:
```
Index(['Ind_Frecuencia_Licor', 'Valor_IMC', 'Ind_Terapia_Hormonal',
       'Ind_Ant_Fam_Otros', 'Ind_Ant_Fam_CAM', 'Edad', 'Ind_CAM',
       'Ind_Raza_Blanca', 'Ind_Menarca_Precoz_12', 'Ind_Menopausia',
       'Ind_Bajo_Peso', 'Ind_Sobrepeso', 'Ind_Obesidad', 'Edad_Log',
       'Edad_Sqrt', 'Ind_Birads_3', 'Ind_Obesidad_Posmenopausia'],
      dtype='object')
```

In [79]:
```
#excl_vars_names_4 = []
#excl_vars_names_4 = ['Ind_Ejercicio','Valor_Colesterol_LDL', 'Valor_Colesterol_Hdl', 'Valor_Colesterol_Total']
excl_vars_names_4 = ['Edad','Edad_Sqrt','Valor_IMC'

                     ]
```

In [80]:
```
pre_model_data.columns
```

Out[80]:
```
Index(['Ind_Frecuencia_Licor', 'Valor_IMC', 'Ind_Terapia_Hormonal',
       'Ind_Ant_Fam_Otros', 'Ind_Ant_Fam_CAM', 'Edad', 'Ind_CAM',
       'Ind_Raza_Blanca', 'Ind_Menarca_Precoz_12', 'Ind_Menopausia',
       'Ind_Bajo_Peso', 'Ind_Sobrepeso', 'Ind_Obesidad', 'Edad_Log',
       'Edad_Sqrt', 'Ind_Birads_3', 'Ind_Obesidad_Posmenopausia'],
      dtype='object')
```

4.3.2. Ejecución

In [81]:
```
pre_model_data = non_excluded_data(data=pre_model_data, excl_vars=excl_vars_names_4)
```

In [82]:
```
pre_model_data.count()
```

Out[82]:
```
Ind_Frecuencia_Licor          106089
Ind_Terapia_Hormonal          106089
Ind_Ant_Fam_Otros             106089
Ind_Ant_Fam_CAM               106089
Ind_CAM                       106089
Ind_Raza_Blanca               106089
Ind_Menarca_Precoz_12         106089
Ind_Menopausia                106089
Ind_Bajo_Peso                 106089
Ind_Sobrepeso                 106089
Ind_Obesidad                  106089
Edad_Log                      106089
Ind_Birads_3                  106089
Ind_Obesidad_Posmenopausia    106089
dtype: int64
```

---

5. Especificación del modelo

Estimación y diagnósticos de un modelo base seleccionado *(regresión logística)*, para analizar la correcta especificación del mismo. El ejercicio se desarrolla con la base completa, al considerar todas las posibles variables explicativas.

5.1. Datos de Modelación

Definición de la base de datos objeto de modelación, para analizar la correcta especificación del modelo base.

5.1.1. Definición de parámetros

In [83]:
```
# Lista con el nombre de las variables independientes a excluir del análisis
#not_includ_vars = ['Ind_Ant_Per_HTA', 'Valor_PAS']
not_includ_vars = []
# Vector con las interaccciones (en forma de fórmula) entre variables independientes
interactions_formula = []
# Diccionario con las transformaciones (en forma de fórmula) por cada variable independiente
    # La clave es el nombre de la variable, y el valor es la transformación.
transform_formula = {}
# Presencia de escalamiento por la media, para las variables numéricas
mean_stand = True
# Presencia de escalamiento por la desviación estándar, para las variables numéricas
std_stand = True
# Nombre de la variable que hace las veces de intercepto para las regresiones.
    # Si no se desea tener el intercepto en la base de datos, se pone: 'None'.
    # De ser considerada, hará parte de la base de datos principal, y será la primer columna.
intercept_name=None
# Acción a ejecutar con las observaciones que representen una categoría nueva, para una variable categórica
    # 'drop': Elimina dichas observaciones
    # 'raise': Saca el nombre del error e interrumpe el proceso
NA_cat_action = 'drop'
# Semilla para generar las particiones de los datos
rdn_state = 123
```

5.1.2. Generación de parámetros adicionales

In [84]:
```
# Exclusión de variables
model_data = non_excluded_data(data=pre_model_data, excl_vars=not_includ_vars)
# Definición de la base preliminar para definir la variable dependiente
model_dep_var = model_data[dep_var_name]
# Definición de la base preliminar para definir las variables independientes
model_ind_vars = model_data.loc[:, model_data.columns != dep_var_name]
```

In [85]:
```
model_data.columns
```

Out[85]:
```
Index(['Ind_Frecuencia_Licor', 'Ind_Terapia_Hormonal', 'Ind_Ant_Fam_Otros',
       'Ind_Ant_Fam_CAM', 'Ind_CAM', 'Ind_Raza_Blanca',
       'Ind_Menarca_Precoz_12', 'Ind_Menopausia', 'Ind_Bajo_Peso',
       'Ind_Sobrepeso', 'Ind_Obesidad', 'Edad_Log', 'Ind_Birads_3',
       'Ind_Obesidad_Posmenopausia'],
      dtype='object')
```

In [86]:
```
# Definición de parámetros adicionales, necesarios para la codificación de los datos
encod_data_params = EncodedDataParams(data=model_data,
                                      dep_var_name=dep_var_name,
                                      cat_ind_vars_def=default_cats)\
    .generateParams()
```

In [87]:
```
encod_data_params
```

Out[87]:
```
{'default_cats': {'Ind_Frecuencia_Licor': 'No',
  'Ind_Terapia_Hormonal': 'No',
  'Ind_Ant_Fam_Otros': 'No',
  'Ind_Ant_Fam_CAM': 'No',
  'Ind_Raza_Blanca': 'N',
  'Ind_Menarca_Precoz_12': 'N',
  'Ind_Menopausia': 'N',
  'Ind_Bajo_Peso': 'N',
  'Ind_Sobrepeso': 'N',
  'Ind_Obesidad': 'N',
  'Ind_Birads_3': 'N',
  'Ind_Obesidad_Posmenopausia': 'N'}}
```

5.1.3. Variable dependiente

Codificación y transformación de la variable dependiente, insumo para todos los procesos que la requieran, de aquí en adelante.

In [88]:
```
y_model_encod = DependEncodedData(dep_var_succ=dep_var_succ_cat, bivar_anal=bivariate_anal)\
    .fit(y=model_dep_var)
y_model_transf = y_model_encod.transform(y=model_dep_var)
```

5.1.4. Variables independientes

Codificación y transformación de las variable independientes. Esta base, sólo será el insumo para el proceso de especificación del modelo.

- **Codificación**

In [89]:
```
prmod_encoded_data = ModelEncodedData(encod_params=encod_data_params,
                                      interac_form=interactions_formula,
                                      transf_form=transform_formula,
                                      with_mean=mean_stand,
                                      with_std=std_stand)
```

In [90]:
```
X_prmod_encod = prmod_encoded_data.fit_transform(X=model_ind_vars)
```

- **Transformación**

In [91]:
```
prmod_transf_data = ModelTransformerData(formula=prmod_encoded_data._X_formula,
                                         const_name=intercept_name,
                                         NA_action=NA_cat_action)
```

In [92]:
```
X_prmod_transf = prmod_transf_data.fit_transform(X=X_prmod_encod)
```

In [93]:
```
X_prmod_transf.count()
```

Out[93]:
```
Ind_Frecuencia_Licor_Si         106089
Ind_Terapia_Hormonal_Si         106089
Ind_Ant_Fam_Otros_Si            106089
Ind_Ant_Fam_CAM_Si              106089
Ind_Raza_Blanca_S               106089
Ind_Menarca_Precoz_12_S         106089
Ind_Menopausia_S                106089
Ind_Bajo_Peso_S                 106089
Ind_Sobrepeso_S                 106089
Ind_Obesidad_S                  106089
Ind_Birads_3_S                  106089
Ind_Obesidad_Posmenopausia_S    106089
Edad_Log                        106089
dtype: int64
```

5.2. Estimación

Estimación del modelo base multivariado, con los complementos de análisis de supuestos, efectos marginales e interpretación de las variables numéricas *(en caso de estar estandarizadas)*.

5.2.1. Definición de parámetros

In [94]:
```
# Añadir a la base un intercepto, siempre y cuando no esté calculado en la base fuente
add_intercept = True
# Número mínimo de elementos únicos para considerar un variable como numérica, en los gráficos de linealidad
min_unique_vals = 0
# Umbral mínimo para considerar multicolinealidad, según el cálculo del VIF (factor de inflación de la varianza)
vif_threshold = 5
```

5.2.2. Objeto principal

In [95]:
```
st = time.time()
prmod_logit_model = LogisticRegressionModelling(dep_var_encod=y_model_transf,
                                                ind_vars_encod=X_prmod_transf,
                                                ind_vars_decod=model_ind_vars,
                                                const_name=intercept_name,
                                                alpha=alpha_num,
                                                min_uniq=min_unique_vals,
                                                vif_thrs=vif_threshold)\
    .generateLogisticResults()

ed = time.time()
print(ed-st)
```

```
8.00390100479126
```

5.2.3. Ejecución

In [96]:
```
st = time.time()
print_logistic_results(logit_obj=prmod_logit_model.logit_results, interact=interact_anal)
```

Estimación del Modelo

Regresión	Resumen. Reg.	Efectos Margin.	Interpret. Vars. Num.

Detalle de la Regresión

Results: Logit

Model:	Logit	Pseudo R-squared:	0.072
Dependent Variable:	Ind_CAM	AIC:	36810.8679
Date:	2024-11-26 16:36	BIC:	36944.8764
No. Observations:	106089	Log-Likelihood:	-18391.
Df Model:	13	LL-Null:	-19822.
Df Residuals:	106075	LLR p-value:	0.0000
Converged:	1.0000	Scale:	1.0000
No. Iterations:	8.0000		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
const	-3.8088	0.0510	-74.6855	0.0000	-3.9087	-3.7088
Ind_Frecuencia_Licor_Si	1.0872	0.0305	35.6596	0.0000	1.0275	1.1470
Ind_Terapia_Hormonal_Si	-0.0291	0.1816	-0.1602	0.8727	-0.3851	0.3269
Ind_Ant_Fam_Otros_Si	0.2981	0.0472	6.3207	0.0000	0.2057	0.3905
Ind_Ant_Fam_CAM_Si	1.3412	0.0908	14.7761	0.0000	1.1633	1.5191
Ind_Raza_Blanca_S	0.4062	0.0406	10.0069	0.0000	0.3267	0.4858
Ind_Menarca_Precoz_12_S	0.1402	0.0316	4.4338	0.0000	0.0782	0.2021
Ind_Menopausia_S	0.2813	0.0657	4.2842	0.0000	0.1526	0.4100
Ind_Bajo_Peso_S	0.1325	0.1375	0.9636	0.3353	-0.1370	0.4020
Ind_Sobrepeso_S	-0.0394	0.0352	-1.1176	0.2637	-0.1084	0.0297
Ind_Obesidad_S	-0.2195	0.0739	-2.9695	0.0030	-0.3643	-0.0746
Ind_Birads_3_S	0.5921	0.0563	10.5230	0.0000	0.4819	0.7024
Ind_Obesidad_Posmenopausia_S	0.2749	0.0815	3.3725	0.0007	0.1151	0.4346
Edad_Log	0.3144	0.0303	10.3750	0.0000	0.2550	0.3738

Análisis de Diagnósticos

Linealidad	VIF (Multicol.)

Linealidad: Log_Odds vs. Variables

In [97]:
```
ed = time.time()
print(ed-st)
```

```
5.35425329208374
```

5.3. Selección recursiva de variables

A través de diferentes métodos, se evalúan distintos conjuntos de variables, y se selecciona aquel que maximiza o minimiza alguna medida de bondad de ajuste, predicción o costo. Este proceso final, se ejecuta en un escenario de validación cruzada.

- Se considera la selección recursiva de variables, como uno de los métodos para controlar por el posible sobre-ajuste de los modelos; sobretodo aquel que se deriva de la presencia de un conjunto numeroso de variables explicativas.

5.3.1. Definición de parámetros

In [98]:
```
# Tipo de análisis empleado para hacer la selección de variables:
    # 'recur': Selección hacia atrás, con el objetivo de maximizar el 'accuracy'.
    # 'sequen': Selección hacia adelante, hacia atrás o de forma híbrida.
    # 'exhaus': Utiliza exahustivamente todas las combinaciones posibles de variables.
    # 'embed': Método de envoltura, que utiliza regularizaciones sobre un modelo base para encontrar coeficientes <> 0.
analysis_type = 'sequen'
# Nombre del modelo a emplear. Se disponen de los siguientes:
    # 'logist': Regresión logística.
    # 'lin_svm': Máquina de soporte vectorial, con kernel lineal.
model_name='logist'
# Nombre de la medida de rendimiento a optimizar.
    # Las medidas disponibles, son todas aquellas que peternecen al módulo: 'metrics', del paquete: 'scikit-learn'
feat_sel_scoring = 'precision'
# Número de particiones para realizar la validación cruzada en el proceso de selección
feat_sel_cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=rdn_state)
# Condición para seleccionar el número de variables, cuando `analysis_type='sequen'`
    # 'best': Escoge el mejor conjunto
    # 'parsimonious': Procura por la parsimonia de los modelos
seq_k_features='best'
# Búsqueda de selección hacia adelante, cuando `analysis_type='sequen'`
seq_forward=True
# Búsqueda de selección hacia atrás, cuando `analysis_type='sequen'`
seq_floating=True
# Número máximo de variables a evaluar en una combinación, cuando `analysis_type='exhaus'`
exh_max_features=len(X_prmod_transf.columns)
# Inverso de la fuerza de regularización, cuando `analysis_type='embed'`
emb_C_param=0.01
# Diccionario para realizar el mapeo de las variables codificadas, al nombre de las variables originales
    # Las claves son el nombre completo de la variable transformada, y el valor, el nombre original de la misma.
encod_dict = prmod_transf_data._encod_data_dict
```

5.3.2. Objeto principal

In [99]:
```
st = time.time()
model_feats_selec = FeaturesSelectors(anal_type=analysis_type,
                                      model=model_name,
                                      const_name=intercept_name,
                                      scoring=feat_sel_scoring,
                                      cv=feat_sel_cv,
                                      k_features=seq_k_features,
                                      forward=seq_forward,
                                      floating=seq_floating,
                                      max_features=exh_max_features,
                                      C_param=emb_C_param,
                                      encod_names=encod_dict)
ed = time.time()
print(ed-st)
```

```
0.0009770393371582031
```

5.3.3. Ejecución

In [100]:
```
st = time.time()
feats_selec_results = model_feats_selec.selectResults(X=X_prmod_transf, y=y_model_transf)
feats_selec_results
```

Out[100]:
```
{'precision': 0.3,
 'select': ['Ind_Frecuencia_Licor_Si',
  'Ind_Terapia_Hormonal_Si',
  'Ind_Ant_Fam_Otros_Si',
  'Ind_Ant_Fam_CAM_Si',
  'Ind_Raza_Blanca_S'],
 'exclud': ['Ind_Menarca_Precoz_12_S',
  'Ind_Menopausia_S',
  'Ind_Bajo_Peso_S',
  'Ind_Sobrepeso_S',
  'Ind_Obesidad_S',
  'Ind_Birads_3_S',
  'Ind_Obesidad_Posmenopausia_S',
  'Edad_Log']}
```

In [101]:
```
ed = time.time()
print(ed-st)
```

```
168.99628067016602
```

5.3.4. Definición de variables finales

Define un lista que contempla el nombre de las variables finales, que entrarán como explicativas en el entrenamiento de modelos de clasificación.

- **Definición de parámetros**

In [102]:
```
# Lista con el nombre completo de las variables que aunque fueron exlcuidas por el análisis, deben estar
X_includ_vars = [
'Ind_Menarca_Precoz_12_S',
  'Ind_Menopausia_S',
  'Ind_Obesidad_S',
  'Ind_Birads_3_S',
  'Ind_Obesidad_Posmenopausia_S',
  'Edad_Log'
]
# Lista con el nombre completo de las variables que aunque no fueron excluidas por el análisis, deben estarlo
X_drop_vars = [

              ]
#X_drop_vars = ['Presion_Sist_Sentado', 'Sexo_M', 'Valor_Creatinina']
```

- **Ejecución**

In [103]:
```
vars_names = model_feats_selec.selectVariables(X_includ=X_includ_vars, X_drop=X_drop_vars)
```

In [104]:
```
vars_names
```

Out[104]:
```
['Edad_Log',
 'Ind_Ant_Fam_CAM',
 'Ind_Ant_Fam_Otros',
 'Ind_Birads_3',
 'Ind_Frecuencia_Licor',
 'Ind_Menarca_Precoz_12',
 'Ind_Menopausia',
 'Ind_Obesidad',
 'Ind_Obesidad_Posmenopausia',
 'Ind_Raza_Blanca',
 'Ind_Terapia_Hormonal']
```

In [105]:
```
# Conjunto de funciones principales
import summary_analysis as descrp
descrp.stat_uncond_descrp_cat_var(col = pre_model_data['Ind_CAM'])
```

Análisis Incondicional

Estadísticos

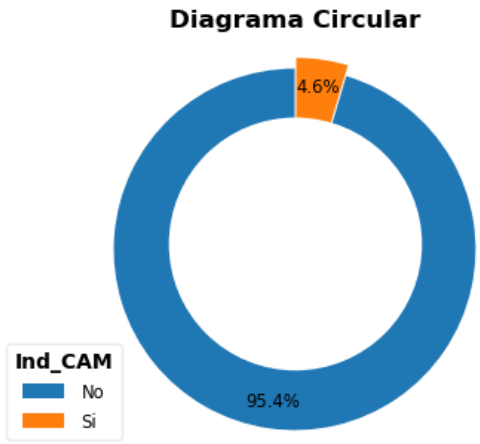
Conteo de Registros

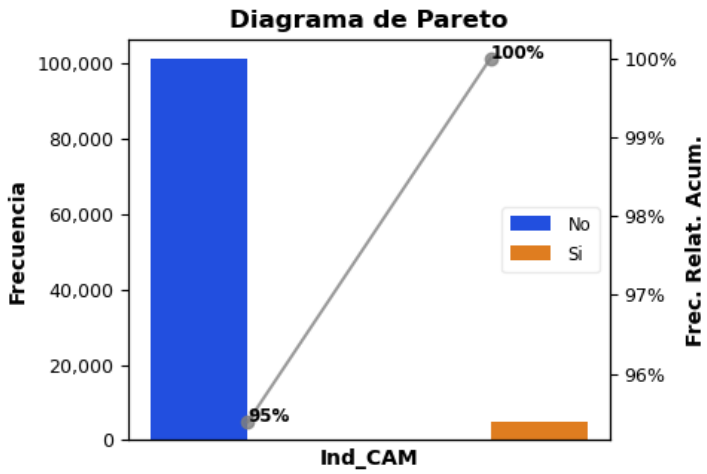
Registros	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Con Dato	106,089	100.00%	100.00%

Conteo de Frecuencias

Categorías	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
No	101,199	95.39%	95.39%
Si	4,890	4.61%	100.00%

Gráficos Descriptivos





5.4. Datos de modelación

Codificación y transformación de las variables independientes, insumo para todos los procesos que la requieran, de aquí en adelante.

5.4.1. Obtención de parámetros

Se obtiene el nombre de las variables originales, según las variables finales del modelo. Además, se definen los nuevos objetos que contemplan las interacciones y transformaciones, según hayan quedado seleccionadas como variables finales.

In [106]:
```
adit_model_params = ModelDataParams(variables=vars_names,
                                    interac_form=interactions_formula,
                                    transf_form=transform_formula)
```

In [107]:
```
X_model_var_names = adit_model_params.var_names()
model_interac_form, model_transf_form = adit_model_params.adit_params()
```

5.4.2. Datos iniciales

Define la base de datos, que contempla las variables finales, definidas en el proceso de selección recursiva.

In [108]:
```
X_model = model_ind_vars[X_model_var_names]
```

5.4.3. Codificación

In [109]:
```
model_encoded_data = ModelEncodedData(encod_params=encod_data_params,
                                      interac_form=model_interac_form,
                                      transf_form=model_transf_form,
                                      with_mean=mean_stand,
                                      with_std=std_stand)
```

In [110]:
```
X_model_encod = model_encoded_data.fit_transform(X=X_model)
```

5.4.4. Transformación

In [111]:
```
model_transf_data = ModelTransformerData(formula=model_encoded_data._X_formula,
                                         const_name=intercept_name,
                                         NA_action=NA_cat_action)
```

In [112]:
```
X_model_transf = model_transf_data.fit_transform(X=X_model_encod)
```

In [113]:
```
y_model_transf
```

Out[113]:
```
659309     0
1387033    0
1918292    0
1071906    0
1241263    0
          ..
2187337    1
2188085    1
2188699    1
2188855    1
2189115    1
Name: Ind_CAM, Length: 106089, dtype: int64
```

5.5. Estimación final

Estimación del modelo base multivariado, con los complementos de análisis de supuestos, efectos marginales e interpretación de las variables numéricas *(en caso de estar estandarizadas)*.

5.5.1. Objeto principal

In [114]:
```
final_logit_model = LogisticRegressionModelling(dep_var_encod=y_model_transf,
                                                ind_vars_encod=X_model_transf,
                                                ind_vars_decod=X_model,
                                                const_name=intercept_name,
                                                alpha=alpha_num,
                                                min_uniq=min_unique_vals,
                                                vif_thrs=vif_threshold)\
    .generateLogisticResults()
```

5.5.2. Ejecución

In [115]:
```
print_logistic_results(logit_obj=final_logit_model.logit_results, interact=interact_anal)
```

Estimación del Modelo

Regresión	Resumen. Reg.	Efectos Margin.	Interpret. Vars. Num.

Detalle de la Regresión

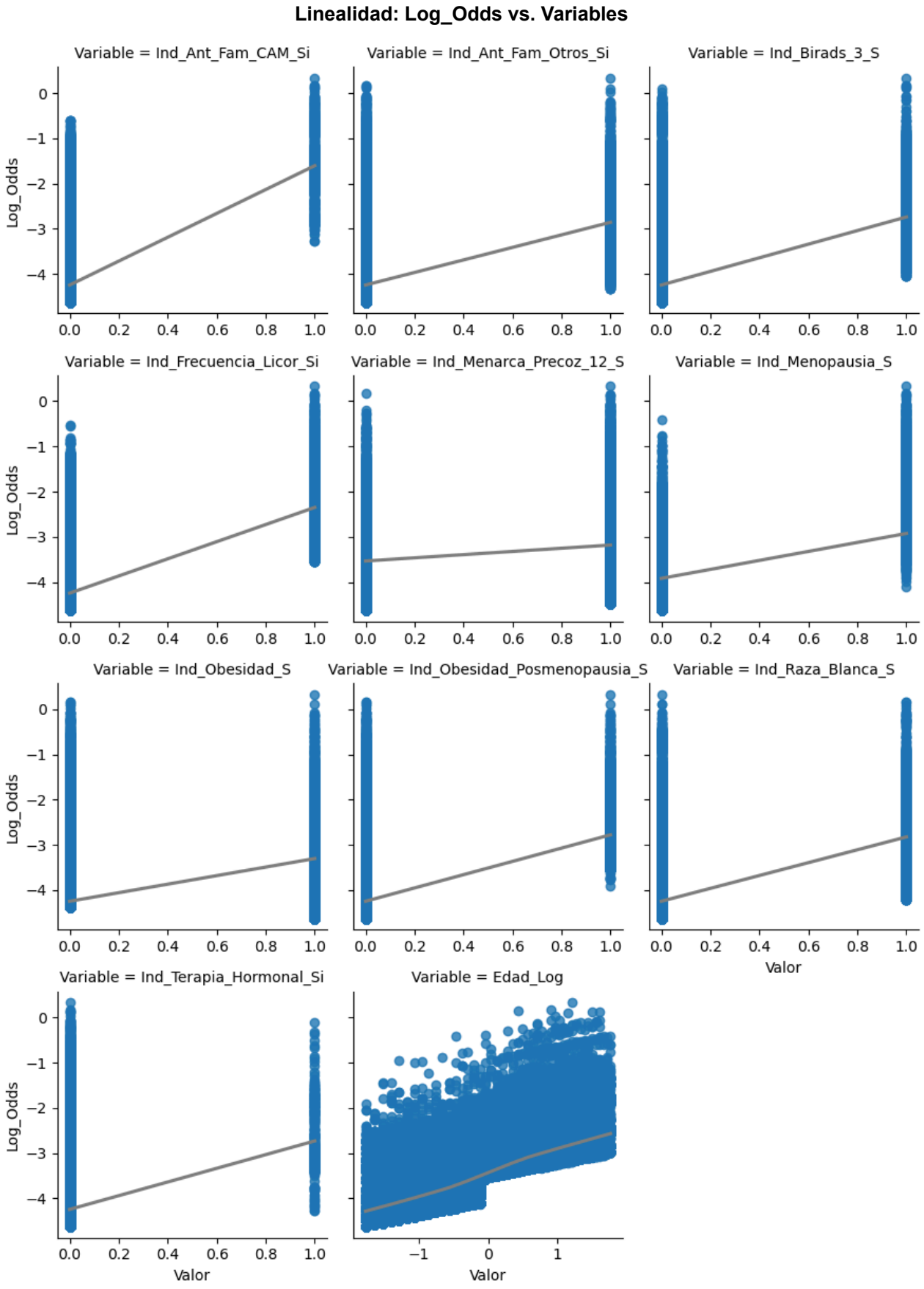
Results: Logit

Model:	Logit	Pseudo R-squared:	0.072
Dependent Variable:	Ind_CAM	AIC:	36809.3422
Date:	2024-11-26 16:39	BIC:	36924.2066
No. Observations:	106089	Log-Likelihood:	-18393.
Df Model:	11	LL-Null:	-19822.
Df Residuals:	106077	LLR p-value:	0.0000
Converged:	1.0000	Scale:	1.0000
No. Iterations:	8.0000		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
const	-3.8251	0.0479	-79.8555	0.0000	-3.9190	-3.7312
Ind_Ant_Fam_CAM_Si	1.3425	0.0908	14.7899	0.0000	1.1646	1.5204
Ind_Ant_Fam_Otros_Si	0.2979	0.0472	6.3164	0.0000	0.2054	0.3903
Ind_Birads_3_S	0.5927	0.0563	10.5343	0.0000	0.4825	0.7030
Ind_Frecuencia_Licor_Si	1.0851	0.0305	35.6291	0.0000	1.0254	1.1448
Ind_Menarca_Precoz_12_S	0.1404	0.0316	4.4402	0.0000	0.0784	0.2023
Ind_Menopausia_S	0.2773	0.0656	4.2302	0.0000	0.1488	0.4058
Ind_Obesidad_S	-0.2407	0.0718	-3.3504	0.0008	-0.3814	-0.0999
Ind_Obesidad_Posmenopausia_S	0.2765	0.0815	3.3937	0.0007	0.1168	0.4362
Ind_Raza_Blanca_S	0.4066	0.0406	10.0163	0.0000	0.3270	0.4861
Ind_Terapia_Hormonal_Si	-0.0287	0.1817	-0.1582	0.8743	-0.3848	0.3274
Edad_Log	0.3159	0.0303	10.4281	0.0000	0.2565	0.3752

Análisis de Diagnósticos

Linealidad | VIF (Multicol.)



6. Modelación

6.1. Definición de muestras

Se definen muestras de entrenamiento y validación, a partir del método habitual: se escoge el porcentaje de datos para alguno de los dos grupos, y usando distintas técnicas de muestreo, se forman los conjuntos de datos mencionados.

- Para modelos de clasificación, donde es interesante conservar los porcentajes de desbalanceo de la muestra, según la variable dependiente, se recomienda utilizar un muestreo aleatorio estratificado.

6.1.1. Definición de parámetros

In [116]:
```
# Proporción del total de registros, destinados para la muestra de validación
test_prop = 0.2
```

6.1.2. Partición

Definición del conjunto de datos de entrenamiento y validación, de las variables independientes y la dependiente. En relación a esta última, ya estaría transformada y lista para modelar.

In [117]:
```
X_train, X_test, y_train_transf, y_test_transf = train_test_split(X_model,
                                                                  y_model_transf,
                                                                  test_size=test_prop,
                                                                  random_state=rdn_state)
```

In [ ]:
```

```

In [ ]:
```

```

6.1.3. Codificación

Codificación o definición del tipo de variables, para el conjunto de predictoras o independientes.

In [118]:
```
training_encoded_data = ModelEncodedData(encod_params=encod_data_params,
                                         interac_form=interactions_formula,
                                         transf_form=transform_formula,
                                         with_mean=mean_stand,
                                         with_std=std_stand)
```

In [119]:
```
X_train_encod = training_encoded_data.fit_transform(X=X_train)
X_test_encod = training_encoded_data.transform(X=X_test)
```

6.1.4. Transformación

Definición de la matriz de diseño de las variables independientes.

In [120]:
```
training_transf_data = ModelTransformerData(formula=training_encoded_data._X_formula,
                                            const_name=intercept_name,
                                            NA_action=NA_cat_action)
```

In [121]:
```
X_train_transf = training_transf_data.fit_transform(X=X_train_encod)
X_test_transf = training_transf_data.transform(X=X_test_encod)
```

6.2. Entrenamiento

Entrenamiento y selección de complejidad de los modelos.

- Para los modelos con hiperparámetros, se selecciona la complejidad en un escenario óptimo de validación cruzada.
- Para todos los modelos, se recolecta los resultados en entrenamiento y validación, para poder construir análisis entorno a las propiedades de calibración y discriminación de los mismos.

6.2.1. Definición de parámetros

In [122]:
```
# Lista, con el nombre de los modelos a entrenar. Los modelos disponibles, son:
    # 'logist': Regresión logística clásica
    # 'logit_class': Regresión logística con ponderación de clases balanceadas
    # 'regul_logist': Regresión logística regularizada
    # 'svm': Máquinas de soporte vectorial (el kernel también será un hiperparámetro a encontrar)
    # 'rf': Bosques aleatorios
    # 'xgb': Arboles de aumento de gradiente (XGBoost)
    # 'nn': Redes neuronales
#models_names = ['rf', 'xgb']
models_names = ['logit_class']
# Medida de rendimiento o costo, utilizada para la optimización de hiperparámetros
    # Las medidas disponibles, son todas aquellas que peternecen al módulo: 'metrics', del paquete: 'scikit-learn'
training_scoring = 'f1'
# Diccionario, con las medidas a maximizar, en torno a la evaluación del mejor modelo
    # La clave corresponde a una abreviación de cada medida, para facilitar la presentación de gráficos
    # El valor corresponde a una de las medidas que pertenecen al módulo: 'metrics', del paquete: 'scikit-learn'
validation_scoring_max = {'avg_prc': 'average_precision',
                          'auc': 'roc_auc',
                          'acc': 'accuracy',
                          'sensit': 'recall',
                          'ppv': 'precision',
                          'bal_acc': 'balanced_accuracy',
                          'f1': 'f1'}
# Diccionario, con las medidas a minimizar, en torno a la evaluación del mejor modelo
    # La clave corresponde a una abreviación de cada medida, para facilitar la presentación de gráficos
    # El valor corresponde a una de las medidas que peternecen al módulo: 'metrics', del paquete: 'scikit-learn'
validation_scoring_min = {'brier_loss':'neg_brier_score'}
```

In [123]:
```
# Método de validación cruzada, para la selección de complejidad de los modelos
training_cv_split = RepeatedStratifiedKFold(n_splits=10, n_repeats=2, random_state=rdn_state)
# Método de validación cruzada, para la validación de los modelos
validation_cv_split = StratifiedKFold(n_splits=10, random_state=rdn_state, shuffle=True)
# Diccionario con la unificación de las medidas a maximizar y minimizar
validation_scoring = {**validation_scoring_max , **validation_scoring_min}
# Lista con las descripciones de las categorías de la variable dependiente, para detalle de la matriz de confusión
y_labels = [*y_model_encod.cats_map]
```

6.2.2. Selección de complejidad

A partir de un escenario de validación cruzada sobre muestra completa, se encuentra la complejidad óptima para cada uno de los modelos, según el promedio de los resultados en muestra de validación y de la medida a optimizar.

In [124]:
```
st = time.time()
fit_models = ClassificationTrainingValidation(models=models_names ,
                                              const_name=intercept_name,
                                              train_scoring=training_scoring ,
                                              train_cv=training_cv_split ,
                                              test_scoring_max=validation_scoring_max,
                                              test_scoring_min=validation_scoring_min,
                                              test_cv=validation_cv_split)\
    .fit(X=X_model_transf, y=y_model_transf)

end = time.time()
print(end-st)
```

```
0.325150728225708
```

6.2.3. Estimaciones adicionales

Se vuelven a estimar los modelos, según la complejidad óptima hallada en el paso anterior, sobre la base de datos de entrenamiento.

- Este proceso es últil cuando se quieran mirar propiedades de predicción en la muestra de validación, una sola vez *(método clásico)*. Lo anterior parte del hecho de que las estimaciones cambian un poco, dependiendo de la muestra que se utilice.

In [125]:
```
fit_train_est, fit_train_dict = fit_models.re_fit(X=X_train_transf, y=y_train_transf)
```

6.3. Validación

La validación de los modelos de clasificación, para variables dependientes dicotómicas, se realiza a partir del análisis de dos conceptos ampliamente utilizados en la literatura:

- **Calibración:** Cualidad de los modelos de clasificación para generar resultados a partir de probabilidades, donde dichos valores predichos pueden ser directamente interpretados como niveles de confianza. Para dicho fin, se emplearán curvas de probabilidad o de calibración, en un contexto de validación cruzada a partir de datos completos.
- **Discriminación:** Capacidad de los modelos para clasificación o discriminar correctamente las observaciones en categorías. Para esto, se tendrán resultados de distintas medidas de rendimiento y costo, en un contexto de validación cruzada a partir de datos completos.

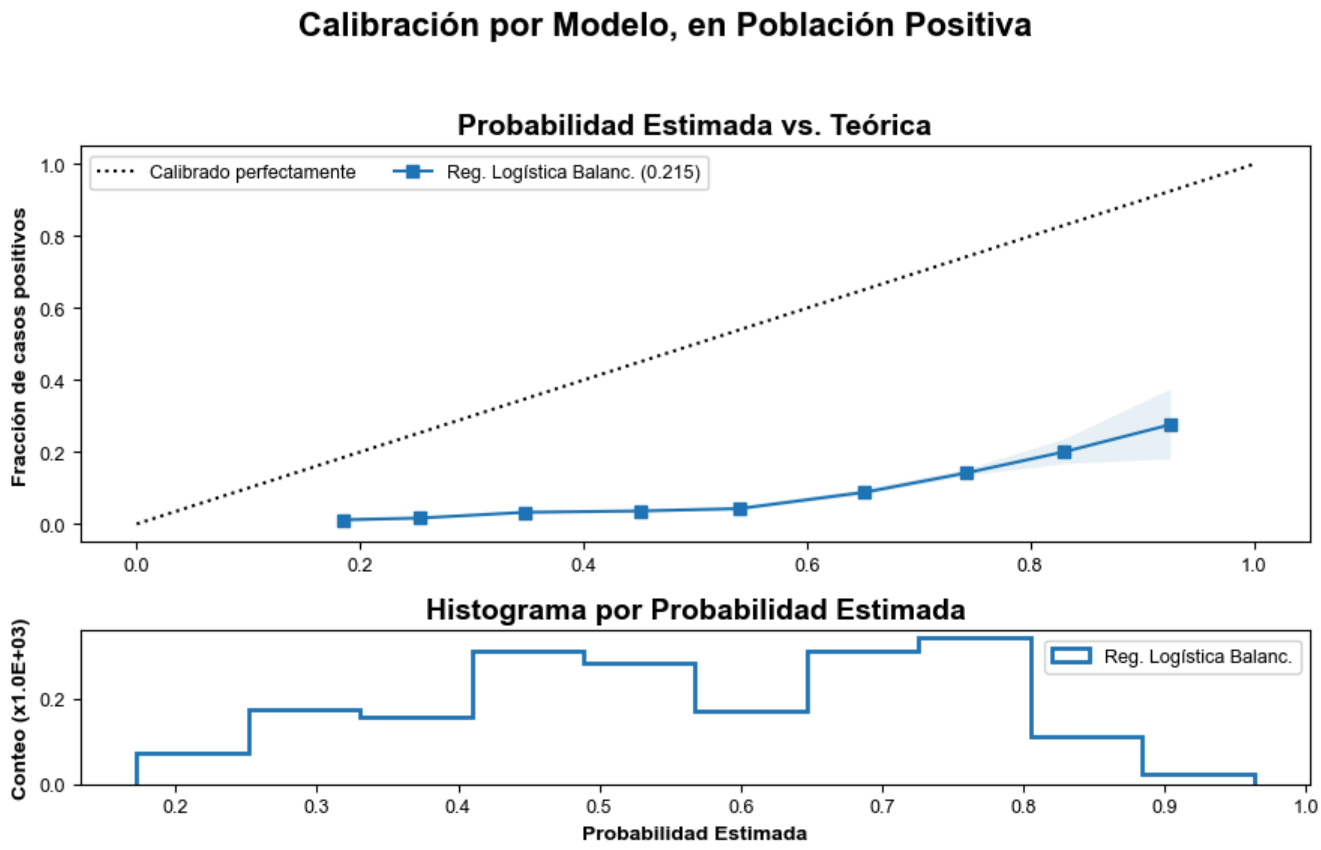
Asimismo, se analizará la composición de varianza y sesgo de los modelos, analizando curvas de aprendizaje, en un contexto de validación cruzada a partir de datos completos.

6.3.1. Curvas de probabilidad

Curvas donde se analiza el promedio y la desviación estándar de la fracción de casos positivos, por intervalos de las probabilidades predichas de la clase positiva, por cada uno de los modelos considerados. Este tipo de curvas se utilizan para mirar la capacidad que tienen los modelos de generar probabilidades de clasificación confiables.

- La banda sobre cada curva, corresponde a una desviación estándar de los resultados en cada muestra de validación.

In [126]:
```
st = time.time()
calibration_curves(models_dict = fit_models.fit_models_dict,
                   X = X_model_transf,
                   y = y_model_transf,
                   cv = validation_cv_split)
```



In [127]:
```
ed = time.time()
print(ed-st)
```

```
4.635570764541626
```

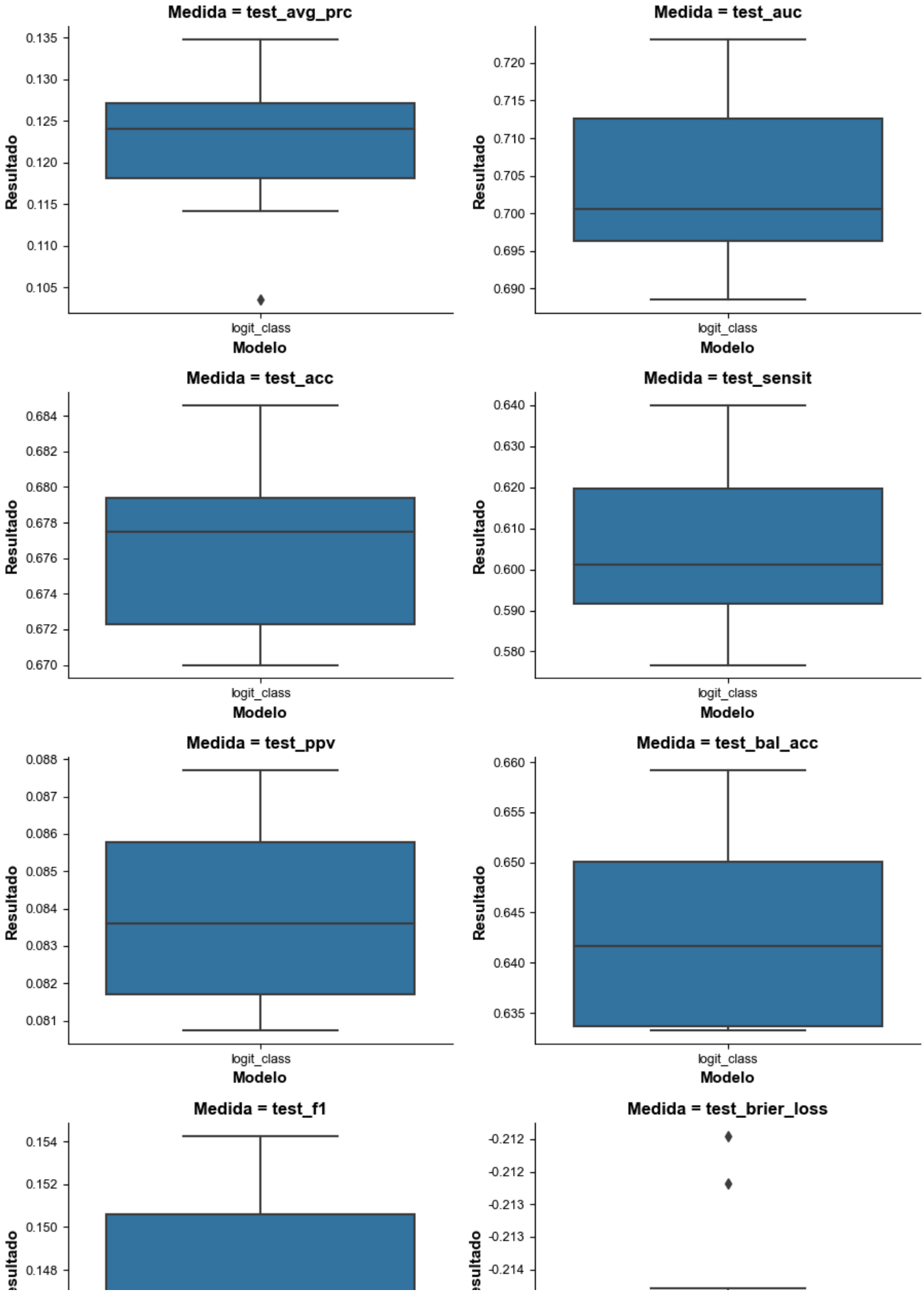
6.3.2. Medidas de costo y rendimiento

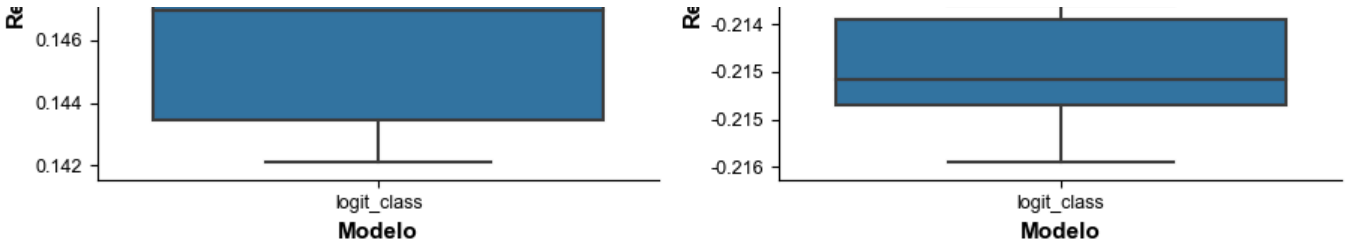
Representación en forma de caja y bigotes, de los resultados en muestra de validación, por medida y modelo complejo. El proceso, es el siguiente:

- Se toma como base los modelos complejos hallados en pasos anteriores y las medidas completas *(para maximizar o minimizar)* de costo y rendimiento.
- A partir del método de validación cruzada seleccionado, se hallan consecutivamente las bases de entrenamiento y validación.
- Se estima cada modelo en la muestra de entrenamiento.
- Se pone a prueba este modelo calibrado en cada muestra de validación.
- Se calculan cada una de las medidas de costo y rendimiento y se almacenan los resultados por cada corrida.
- Se grafican dichos resultados por medida y modelo.

In [128]:
```
st = time.time()
cv_scoring_graphs(models_dict=fit_models.fit_models_dict,
                  X=X_model_transf,
                  y=y_model_transf,
                  cv=validation_cv_split,
                  score=validation_scoring)
```

Rendimiento en Validación Cruzada





In [129]:
```
ed = time.time()
print(ed-st)
```

```
6.119450569152832
```

6.3.3. Curvas de aprendizaje

Curvas que muestran los resultados de algunas métricas, en muestra de entrenamiento y validación, para distintos tamaños de observaciones. Esta herramienta es útil para observar el beneficio de añadir más información y para analizar la composición entre sesgo y varianza de un estimador o modelo determinado.

- **Definición de parámetros**

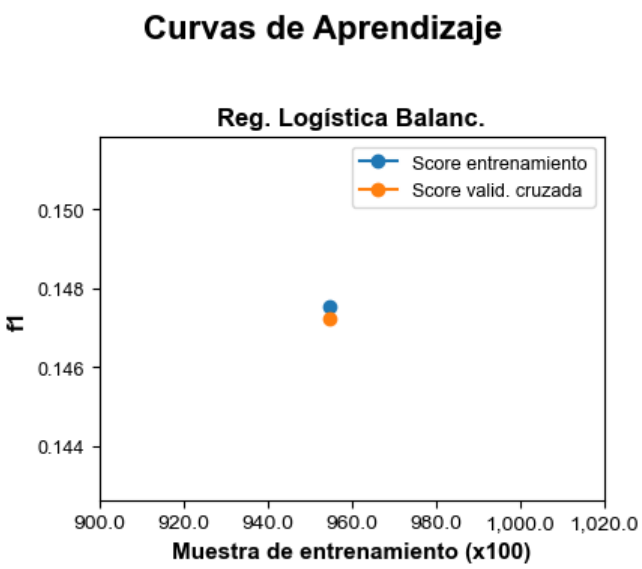
In [130]:
```
# Tupla que define los puntos máximos y mínimos, de la medida de rendimiento a graficar.
    # En caso de dejarle al proceso que encuentre estos valores, se define el párametro con el valor: None
lc_ylim = None
# Vector numérico con las proporciones de la muestra total a considerar en cada corrida del modelo
sampling_size = np.linspace(.1, 1.0, 10)
# Número de procesadores a utilizar en el proceso
    # En caso de querer utilizar todos los procesadores disponibles, se define el párametro con el valor: -1
lc_njobs = -1
# Tupla con el tamaño en pixeles de cada gráfico (horizontal, vertical)
lc_figsize = (4.5, 4)
```

- **Ejecución**

In [131]:
```
st = time.time()
learning_curves(models_dict = fit_models.fit_models_dict,
                X = X_model_transf,
                y = y_model_transf,
                ylim = lc_ylim,
                train_sizes = sampling_size,
                cv = validation_cv_split,
                score = training_scoring,
                n_jobs = lc_njobs,
                figsize = lc_figsize)

ed = time.time()
print(st-ed)
```

```
-10.538529634475708
```



6.4. Selección del mejor modelo

Análisis de agregación de los resultados por modelo en validación cruzada. El proceso es el siguiente:

- Se calcula una medida de agregación de los resultados en muestra de validación, por medida y modelo. Por el momento, se consideró la mediana, como una buena métrica del valor más probable, ante distribuciones sesgadas.
- Según la medida selecciona, se calcula el máximo o el mínimo de los resultados.

6.4.1. Resultados

Tabla con el nombre de los modelos que maximizan o minimizan la mediana de cada medida seleccionada.

In [132]:
```
st = time.time()
scoring_best_models(models_dict=fit_models.fit_models_dict,
                    X=X_model_transf,
                    y=y_model_transf,
                    cv=validation_cv_split,
                    scoring_max=validation_scoring_max,
                    scoring_min=validation_scoring_min)
```

Out[132]:

		Resultado	Tipo
Medida	**Modelo**		
acc	**logit_class**	0.68	Máximo
auc	**logit_class**	0.70	Máximo
avg_prc	**logit_class**	0.12	Máximo
bal_acc	**logit_class**	0.64	Máximo
f1	**logit_class**	0.15	Máximo
ppv	**logit_class**	0.08	Máximo
sensit	**logit_class**	0.60	Máximo
brier_loss	**logit_class**	-0.22	Mínimo

In [133]:
```
ed = time.time()
print(ed-st)
```

```
4.967820644378662
```

6.4.2. Selección

Selecciona el mejor modelo de clasificación, junto con su complejidad, y se estima sobre la base de datos completa. **Este objeto será el modelo final, insumo para la producción en la predicción del fenómeno estudiado.**

- **Definición de parámetros**

In [134]:
```
# Nombre del modelo, seleccionado como el mejor
    # Si el parámetro es 'None', el modelo será aquel con los mejores resultados según la medida usada para la complejidad
best_model_name = 'logit_class'
#best_model_name = 'logist'
```

- **Ejecución**

In [135]:
```
st = time.time()
best_model = fit_models.fit_model_est(model_name=best_model_name)\
    .fit(X=X_model_transf, y=y_model_transf)

ed = time.time()
print(ed-st)
```

```
0.3591465950012207
```

6.5. Selección de puntos de corte

Se selecciona el punto de corte de probabilidad para el mejor modelo. Para lograr este objetivo, se han implementado algunas aproximaciones metodológicas, entre las que se encuentran:

- Curvas ROC en un escenario de validación cruzada.
- Curvas Sensibilidad-Precisión en un escenario de validación cruzada.
- Principales medidas resultado de distintas matrices de confusión, en un escenario de validación cruzada.
- Análisis de la frecuencia absoluta y porcentual de los clasificados positivos y negativos, tomando como insumo la base de datos completa.
- Optimización de funciones definidas positivas y negativas, que dependen del punto de corte.

Por último, vale la pena enunciar que esta aproximación, solo se ejecutará para aquellos tipos de modelos cuyo implemento de decisión en la clasificación, sea una probabilidad, o una variable semejante.

6.5.1. Definición de parámetros

In [136]:
```
# Lista con el nombre de las medidas bajo las cuales se obtendrá un corte óptimo. Se disponen de las siguientes:
    # 'eq_sens_spec': Diferencia entre sensibilidad y especificidad -> Mínimo
    # 'mct': Error de clasificación, con costos diferenciales entre un FN y un FP -> Mínimo
    # 'yi': Índice de Youden o distancia de la curva ROC a cada punto de la curva de aleatoriedad -> Máximo
    # 'f1': Media armónica entre la sensibilidad y la precisión -> Máximo
    # 'bal_acc': Accuracy balanceado o media entre la sensibilidad y especificidad -> Máximo
    # 'roc_dist': Distancia de la curva ROC al punto de rendimiento perfecto -> Mínimo
    # 'pr_dist': Distancia de la curva Precisión - Sensibilidad, al punto de rendimiento perfecto -> Mínimo
cutoff_meas = ['eq_sens_spec', 'mct', 'yi', 'f1', 'bal_acc', 'roc_dist', 'pr_dist']
# Número de puntos de corte candidatos para la elección del óptimo
thrs_num = 100
# Número real que define el factor escala para los conteos de personas en la división de predicciones
count_scale = 1e3
```

In [137]:
```
# Costo de un falso negativo
cutoff_cfn = model_dep_var.value_counts().sum() / model_dep_var.value_counts()[dep_var_succ_cat]
# Costo de un falso positivo
cutoff_cfp = 1
```

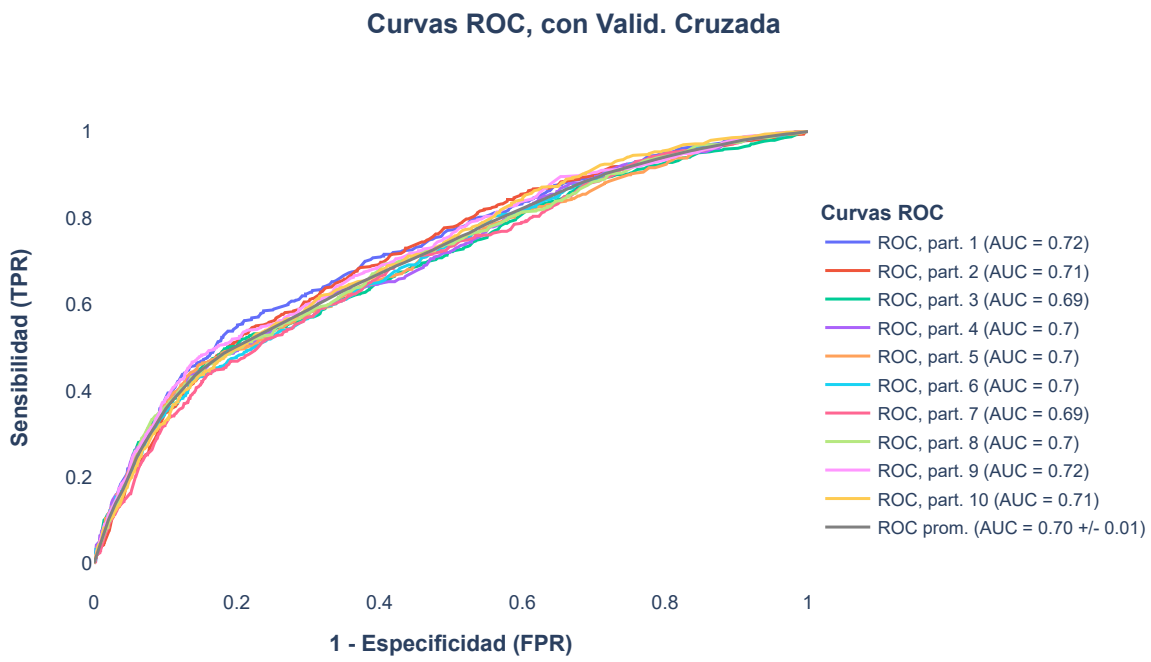
6.5.2. Ejecución

In [138]:
```
st = time.time()
print_cut_offs_selection(estimator=best_model,
                         X=X_model_transf,
                         y=y_model_transf,
                         cv=validation_cv_split,
                         meas=cutoff_meas,
                         cfn=cutoff_cfn,
                         cfp=cutoff_cfp,
                         thrs_num=thrs_num,
                         count_scale=count_scale,
                         interact=interact_anal)

ed = time.time()
print(ed-st)
```

▾ **Gráficos de diagnóstico**

Curvas ROC | Curvas PR | Curvas Mat. Conf. | Poblaciones Pred.



▾ **Puntos óptimos**

Resultados | Gráficos

Puntos de corte óptimos por medida

Función	Corte (Thr)	Valor	Tipo
eq_sens_spec	0.49	0.01	Mínimo
mct	0.57	0.69	Mínimo
yi	0.57	0.30	Máximo
f1	0.68	0.21	Máximo
bal_acc	0.57	0.65	Máximo
roc_dist	0.50	0.51	Mínimo
pr_dist	0.30	0.95	Mínimo

```
242.66506958007812
```

6.5.3. Selección

Definición del punto de corte, en base a los análisis previamente expuestos o criterio de expertos.

In [139]:
```
cut_off_value = 0.49
```

6.6. Reporte de Clasificación

Reporte dinámico de clasificación *(se puede cambiar el punto de corte de forma interactiva)*, a partir de la selección del punto de corte óptimo para el mejor modelo. Para este caso, se utiliza el método clásico de partición *(una sola división entre datos de entrenamiento y validación)*, en lugar del método de validación cruzada.

Los informes presentados en orden, son:

- Principales medidas de rendimientos, en forma de reporte.
- Matriz de confusión completa.
- Densidades de valores predichos o de las probabilidades estimadas, condiciones por la condición verdadera de clasificación.

In [140]:
```
X_test_transf.columns
```

Out[140]:
```
Index(['Ind_Ant_Fam_CAM_Si', 'Ind_Ant_Fam_Otros_Si', 'Ind_Birads_3_S',
       'Ind_Frecuencia_Licor_Si', 'Ind_Menarca_Precoz_12_S',
       'Ind_Menopausia_S', 'Ind_Obesidad_S', 'Ind_Obesidad_Posmenopausia_S',
       'Ind_Raza_Blanca_S', 'Ind_Terapia_Hormonal_Si', 'Edad_Log'],
      dtype='object')
```

In [141]:
```
opt_cutoff_class_report(estimator=fit_train_est[best_model_name],
                        X_test=X_test_transf,
                        y_test=y_test_transf,
                        cut_off=cut_off_value,
                        y_labels=y_labels,
                        interact=interact_anal)
```

☑ Punto óptimo

Umbral 0.49

Ejecución

Reporte	Matriz	Prob. Estim.

Reporte de Clasificación

	precision	recall	f1-score	support
No	0.97	0.64	0.77	20243
Si	0.08	0.66	0.14	975
accuracy			0.64	21218
macro avg	0.53	0.65	0.46	21218
weighted avg	0.93	0.64	0.74	21218

In [142]:
```
#cond1 = (X_test['Edad_Sqrt']**2>=65)&(X_test['Ind_Menarca_Precoz_12']=='N')&(X_test['Ind_Raza_Blanca']=='N')
#cond2 = (X_test['Edad_Sqrt']**2>=65)&(X_test['Ind_Menarca_Precoz_12']=='S')&(X_test['Ind_Raza_Blanca']=='N')
#cond3 = (X_test['Edad_Sqrt']**2<65)&(X_test['Ind_Menarca_Precoz_12']=='N')&(X_test['Ind_Raza_Blanca']=='S')
#
#
#indexes = X_test[(cond1)|(cond2)|(cond3)].index.tolist()
```

In [143]:
```
#X_test_2 = X_test.copy()
#
#neg_df = X_test_2.loc[~X_test_2.index.isin(neg_ind)]
#pos_df = X_test_2.loc[X_test_2.index.isin(neg_ind)]
#
#
#pos_df['Edad'] = np.select(
#    [
#        pos_df['Edad']>65
#    ],
#    [
#        65
#    ],
#    default=pos_df['Edad']
#)
#
#pos_df['Ind_Menarca_Precoz_12']='N'
#pos_df['Ind_Raza_Blanca']='N'
#
#X_test_encod_2 = training_encoded_data.transform(X=pos_df.append(neg_df))
#X_test_transf_2 = training_transf_data.transform(X=X_test_encod_2)
#
#print(opt_cutoff_class_report(estimator=fit_train_est[best_model_name],
#                          X_test=X_test_transf_2,
#                          y_test=y_test_transf,
#                          cut_off=cut_off_value,
#                          y_labels=y_labels,
#                          interact=interact_anal)
#     )
```

7. Agrupamiento de probabilidades

.Agrupamiento de observaciones semejantes, en relación a la probabilidad de pertenencia a una categoría de éxito. El objetivo de este ejercicio radica en poder visualizar desde grupos, observaciones cuyas características en materia de probabilidad, son semejantes.

- Al constar de un análisis donde la única variable para generar el agrupamiento, es numérica real *(probabilidad)*, se emplea el algoritmo K-means como modelo por defecto.

7.1. Definición de parámetros

Parámetros para construir los grupos y visualizar correctamente los resultados.

In [144]:
```
# Nombre de la variable resultante de la estratificación o agrupamiento
strat_var_name = 'Estratificación'
# Parámetro lógico que indica si la condición de éxito será la definida por el parámetro: `dep_var_succ_cat`
    # Si `inv_score = False`, se tomará la probabilidad de pertenecer a la categoría del parámetro: `dep_var_succ_cat`
    # Si `inv_score = True`, se tomará la probabilidad de no pertenecer a la categoría del parámetro: `dep_var_succ_cat`
inv_score = False
# Lista con las categorías de éxito, en orden ascendente (según la probabilidad utilizada, a partir del punto de corte)
    # Si `inv_score = False`, la escala irá de menor a mayor éxito
    # Si `inv_score = True`, la escala irá de mayor a menor éxito
risk_labels = ['Alto', 'Muy Alto']
#risk_labels = ['Bajo', 'Medio']
# Lista con las categorías de fracaso, en orden ascendente (según la probabilidad utilizada, a partir del punto de corte)
    # Si `inv_score = False`, la escala irá de menor a mayor éxito
    # Si `inv_score = True`, la escala irá de mayor a menor éxito
risk_free_labels = ['Bajo', 'Medio']
#risk_free_labels = ['Alto', 'Muy Alto']
#risk_free_labels = ['Alto']
```

7.2. Ejecución

7.2.1. Probabilidad predicha

Calculo de la predicción de la probabilidad para todas las observaciones, según la definición de categoría de éxito para este punto de agrupamiento.

In [145]:
```
y_model_score = _y_pred_prediction_method(fit_est=best_model, X_test=X_test_transf, inv_score=inv_score)
```

In [146]:
```
outdf = X_test.copy()
outdf['Score'] = y_model_score
outdf['Actual'] = y_test_transf
```

In [147]:
```
#outdf[outdf['Score'].between(cut_off_value,cut_off_value+0.2)]
```

7.2.2. Calibración y predicción

Calibración y predicción del modelo de agrupamiento, haciendo uso de los parámetros especificados previamente y tomando como insumo la base de datos completa de modelación.

In [148]:
```
ind_risk_strat = IndividualRiskStratification(cut_off=cut_off_value,
                                              inv_score=inv_score,
                                              risk_labs=risk_labels,
                                              risk_free_labs=risk_free_labels,
                                              rnd_state=rdn_state)
```

In [149]:
```
ind_strat_res = ind_risk_strat.fit_predict(y_score=y_model_score)
```

In [150]:
```
outdf['Strat'] = ind_strat_res
```

Definición de regla de negocio para reducir la población a gestionar, la cual correspondería a las estratificaciones 'Alto' y 'Muy Alto': para ello consideramos la probabilidad más pequeña donde se cumplen las condiciones de antecedente familiar y birads 3. La categoría 'Alto' pasa a 'Medio'. Posteriormente dividimos la estratificación 'Muy Alto'. Para el caso donde se supera la probabilidad umbral se preserva la categoría 'Muy Alto', para el caso donde la probabilidad es inferior se baja a la categoría 'Alto'.

7.3. Visualización

Tablas y gráficos que muestran los resultados de estratificación o agrupamiento de las probabilidades predichas. El ejercicio se realiza bajo escenarios incondicionales y condicionales *(según la variable dependiente o de clasificación)*.

In [151]:
```
print_risk_stratification(y=model_dep_var, y_score=y_model_score, strat=ind_strat_res,
                          cent_labs=ind_risk_strat._labels_centers, risk_labs=risk_labels, risk_free_labs=risk_free_labels,
                          idx_name=strat_var_name, inv_score=inv_score, interact=interact_anal)
```

▾ **Tablas resumen**

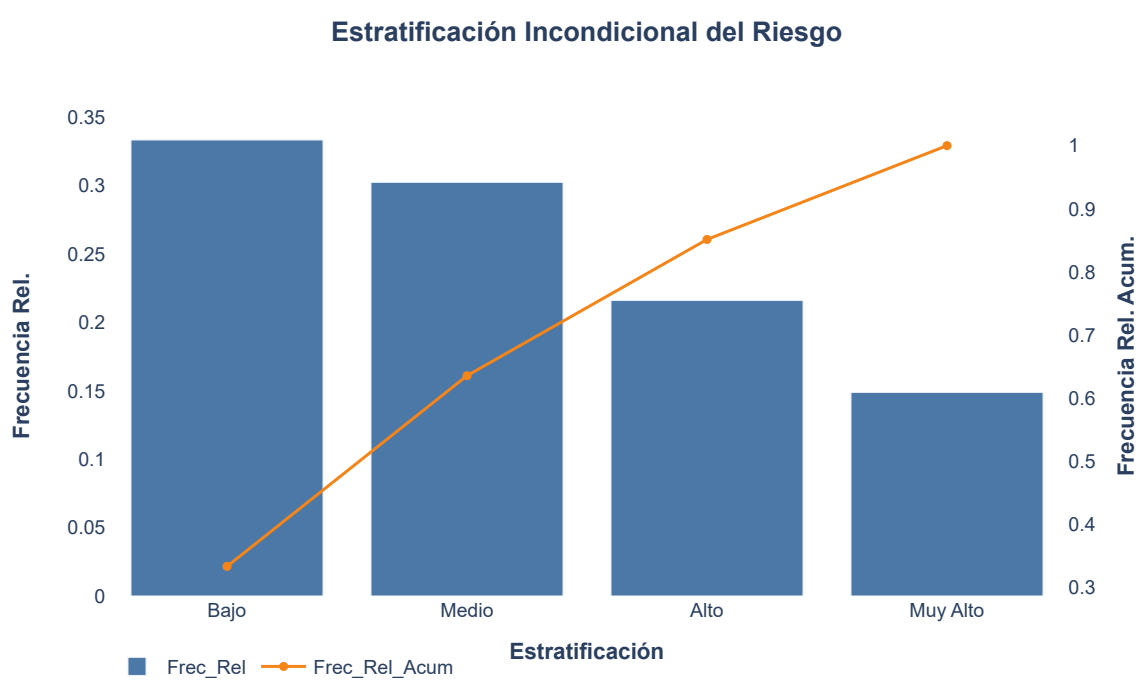
Estrat. Incond. | Interv. Estrat. | Estrat. Cond.

Estratificación incondicional

Estratificación	Frec_Abs	Frec_Rel	Frec_Rel_Acum
Bajo	7,070	33.32%	33.32%
Medio	6,412	30.22%	63.54%
Alto	4,581	21.59%	85.13%
Muy Alto	3,155	14.87%	100.00%

▾ **Gráficos descriptivos**

Estrat. Incond. | Distr. Incond. | Estrat. Cond.



In [ ]:
```
```

8. Almacenamiento

Almacenamiento del objeto necesario para completar el proceso de producción del fenómeno modelado.

8.1. Definición de parámetros

8.1.1. Rutas y nombres de archivos

In [152]:
```
# Ruta relativa (a partir del Notebook de ejecución) donde se almacenará el objeto resultado
output_dir = "../ops/objects/"
# Sufijo para el nombre del objeto resultado
prefix_name = "model"
# Formato del objeto resultado
format_ = ".pkl"
```

In [153]:
```
# Nombre del archivo sobre el cual quedará guardado el objeto resultado
output_file_name = output_dir + prefix_name + '_' + main_name + format_
```

In [154]:
```
output_file_name
```

Out[154]:
```
'../ops/objects/model_cam.pkl'
```

8.1.2. Objetos adicionales

- **Modelo final**

Mejor modelo, calibrado a partir de los datos completos.

In [155]:
```
final_model = fit_models.fit_model_est(model_name=best_model_name)
final_model.fit(X=X_model_transf, y=y_model_transf)
```

Out[155]:
```
LogisticRegression(class_weight='balanced', penalty='none')
```

- **Transformación de variables**

Objeto destinado para la transformación de las variables finales, sin ser estimado, para que se pueda almacenar en memoria *(hay algunos elementos de los objetos `patsy`, que no se pueden almacenar)*.

In [156]:
```
# Objeto destinado para la transformación de las variables finales, sin ser estimado
output_transf_data = ModelTransformerData(formula=model_encoded_data._X_formula,
                                          const_name=intercept_name,
                                          NA_action=NA_cat_action)
```

In [157]:
```
model_encoded_data._X_formula
```

Out[157]:
```
'Edad_Log + Ind_Ant_Fam_CAM + Ind_Ant_Fam_Otros + Ind_Birads_3 + Ind_Frecuencia_Licor + Ind_Menarca_Precoz_12 + Ind_Menopausia + Ind_Obesidad + Ind_Obesidad_Posmenopausia + Ind_Raza_Blanca + Ind_Terapia_Hormonal'
```

8.1.3. Imputación de variables

Espacio reservado para la ejecución de un proceso simple de imputación de variables, donde se pueden utilizar funciones prediseñadas del paquete: `scikit-learn`, a nivel de variable o de grupos de las mismas.

- **Definición de parámetros**

In [158]:
```
X_model_var_names
```

Out[158]:
```
['Edad_Log',
 'Ind_Ant_Fam_CAM',
 'Ind_Ant_Fam_Otros',
 'Ind_Birads_3',
 'Ind_Frecuencia_Licor',
 'Ind_Menarca_Precoz_12',
 'Ind_Menopausia',
 'Ind_Obesidad',
 'Ind_Obesidad_Posmenopausia',
 'Ind_Raza_Blanca',
 'Ind_Terapia_Hormonal']
```

In [159]:
```
#Ind_Ant_Familiar_Si
#Ind_Frecuencia_Licor_Si
#Ind_Menarca_Precoz_12_S
#Ind_Ooforectomia_Bilateral_Si
#Ind_Raza_Blanca_S
#Ind_Terapia_Hormonal_Si
```

In [160]:
```
# Valor lógico que determina si el proceso de imputación se hará por tipo de variable (numéricas y categóricas) o por variab
column_selector = False
# Diccionario con la especificación de la imputación de las variables. Depende del valor del parámetro `column_selector`, as
    # 'True':
        # - Las claves corresponden a: 'cat', para las variables categóricas, o 'num', para las variables numéricas
        # - En caso de que no se quiera imputar algún subconjunto de variables, se pone la palabra: 'passthrough'
    # 'False':
        # - Las claves corresponden al nombre de las variables que se van a imputar y el valor, al modelo a emplear
        # - En caso de que no se quiera imputar alguna variable, no se pone o en el valor se pone la palabra: 'passthrough'
imput_specification = {
    'Ind_Frecuencia_Licor': SimpleImputer(missing_values=None, strategy='constant', fill_value='No'),
    'Ind_Terapia_Hormonal': SimpleImputer(missing_values=None, strategy='constant', fill_value='No'),
    'Ind_Ant_Fam_Otros': SimpleImputer(missing_values=None, strategy='constant', fill_value='No'),
    'Ind_Ant_Fam_CAM': SimpleImputer(missing_values=None, strategy='constant', fill_value='No'),
    'Ind_Raza_Blanca': SimpleImputer(missing_values=None, strategy='constant', fill_value='N'),
    'Ind_Menarca_Precoz_12': SimpleImputer(missing_values=None, strategy='constant', fill_value='N'),
    'Ind_Menopausia': SimpleImputer(missing_values=None, strategy='constant', fill_value='N'),
    'Ind_Obesidad_Posmenopausia': SimpleImputer(missing_values=None, strategy='constant', fill_value='N'),
    'Ind_Obesidad': SimpleImputer(missing_values=None, strategy='constant', fill_value='N'),
    'Ind_Birads_3': SimpleImputer(missing_values=None, strategy='constant', fill_value='N')
}
```

In [161]:
```
imput_data = df.loc[df[dep_var_name] != dep_var_succ_cat, :][X_model_var_names]
```

- **Ejecución**

In [162]:
```
imput_model = ImputationMissingValues(var_names=X_model_var_names,
                                      column_sel=column_selector,
                                      imput_spec=imput_specification).fit(X=imput_data)
```

- **Resultados**

In [163]:
```
imp_result = pd.DataFrame(data= [str(y) for x in [list(val.statistics_)
                                                  for key, val in imput_model.preprocessor.named_transformers_.items()
                                                  if not isinstance(val, str)]
                                 for y in x],
                          columns = ['Valor_Imp'],
                          index = pd.Index(data=imput_model.col_names,
                                           name='Variable'))
imp_result = widgets.HTML(imp_result.style\
                          .set_table_attributes('class="table table-striped"')\
                          .render())
imp_result
```

Variable	Valor_Imp
Ind_Frecuencia_Licor	No
Ind_Terapia_Hormonal	No
Ind_Ant_Fam_Otros	No
Ind_Ant_Fam_CAM	No
Ind_Raza_Blanca	N
Ind_Menarca_Precoz_12	N
Ind_Menopausia	N
Ind_Obesidad_Posmenopausia	N
Ind_Obesidad	N
Ind_Birads_3	N

8.1.4. Objeto final

Dicho objeto contiene en sí mismo la siguiente lista de objetos *(en su orden respectivo)*:

- Lista con el nombre de las variables finales requeridas por el modelo seleccionado.
- Objeto destinado para la transformación de las variables finales, hasta el punto de llegar a una base de datos correctamente codificada. Es decir, a partir de este objeto, las variables numéricas pueden ser estandarizadas según parámetros de los datos de calibración, y las variables categóricas son definidas como tales, junto con sus categorías por defecto.
- Objeto utilizado para pasar de una base de datos codificada, a una matriz de diseño con variables numéricas. Esta matriz final incluye las interacciones y transformaciones de las variables, consideradas en la etapa de calibración de modelos. Además, para las variables categóricas, el objeto contempla las acciones a realizar cuando aparezcan categorías que no se presentaron en la calibración.
- Modelo final, junto con su complejidad encontrada, estimado con la base de datos completa. Este será el objeto principal para que una vez se tengan las variables pertinentes, se puedan generar las probabilidades predichas de éxito, en la etapa de producción.
- Parámetro lógico que indica si la condición de éxito será la definida por el parámetro: `dep_var_succ_cat`.
- Objeto para generar las agrupaciones finales de las probabilidades predichas, a partir de la calibración de un K-means y la articulación otra serie de parámetros.

In [164]:
```
prod_object = {'X_vars': X_model_var_names,
               'encod': model_encoded_data,
               'transf': output_transf_data,
               'model': final_model,
               'inv_score': inv_score,
               'strat': ind_risk_strat,
               'imput': imput_model}
```

8.2. Ejecución

In [165]:
```
save_object(obj=prod_object, filename=output_file_name)
```

In [166]:
```
prod_object
```

Out[166]:
```
{'X_vars': ['Edad_Log',
  'Ind_Ant_Fam_CAM',
  'Ind_Ant_Fam_Otros',
  'Ind_Birads_3',
  'Ind_Frecuencia_Licor',
  'Ind_Menarca_Precoz_12',
  'Ind_Menopausia',
  'Ind_Obesidad',
  'Ind_Obesidad_Posmenopausia',
  'Ind_Raza_Blanca',
  'Ind_Terapia_Hormonal'],
 'encod': <data_definition.ModelEncodedData at 0x200363adc88>,
 'transf': <data_definition.ModelTransformerData at 0x2006b820d48>,
 'model': LogisticRegression(class_weight='balanced', penalty='none'),
 'inv_score': False,
 'strat': <risk_stratification.IndividualRiskStratification at 0x2006abe79c8>,
 'imput': <imputation.ImputationMissingValues at 0x2006b826508>}
```

In [167]:
```
pre_model_data.head(2)
```

Out[167]:

	Ind_Frecuencia_Licor	Ind_Terapia_Hormonal	Ind_Ant_Fam_Otros	Ind_Ant_Fam_CAM	Ind_CAM	Ind_Raza_Blanca	Ind_Menarca_Precoz_12	Ind_M
659309	No	No	No	No	No	N	S	
1387033	No	No	No	No	No	N	N	

Con todos los datos

In [168]:
```
pd_imput_data = imput_model.transform(X=X_test[X_model_var_names])
```

In [169]:
```
X_output_transf = output_transf_data.fit_transform(model_encoded_data.transform(pd_imput_data))
```

In [170]:
```
# Probabilidad de ocurrencia del evento seleccionado
_probs = _y_pred_prediction_method(fit_est = final_model,
                                   X_test = X_output_transf,
                                   inv_score = inv_score)
# Probabilidad de ocurrencia del evento seleccionado
_inv_probs = _y_pred_prediction_method(fit_est = final_model,
                                   X_test = X_output_transf,
                                   inv_score = not inv_score)
```

In [171]:
```
# Estratificaciones de la probabilidad calculada
_strats = ind_risk_strat.predict(y_score = _probs)

_inv_strats = ind_risk_strat.predict(y_score = _inv_probs)
```

In [172]:
```
print_risk_stratification(y=model_dep_var, y_score=_probs, strat=_strats,
                          cent_labs=ind_risk_strat._labels_centers, risk_labs=risk_labels, risk_free_labs=risk_free_labels,
                          idx_name=strat_var_name, inv_score=inv_score, interact=interact_anal)
```

```
Output()
```

Corrida en toda la poblacion

In [173]:
```
import pickle

DB = "indice_desenlaces"
OPS_TB_NAME = "dm_data"
PIPELINE_PATH = r"C:/Users/juan.echeverri/Documents/SURA/Indice_Salud/Indice_Salud/events/ops/objects/model_v2_cm.pkl"
PARAMS_SEP = ','

with open(PIPELINE_PATH,'rb')as f:
  modelo_cm = pickle.load(f)
```

In [174]:
```
X_model_var_names =  modelo_cm["X_vars"]
X_model = df[X_model_var_names]
X_model['Ind_Frecuencia_Licor'] = X_model['Ind_Frecuencia_Licor'].fillna('No')
```

In [175]:
```
vars_ = modelo_cm['X_vars']
df = df[vars_]
```

In [176]:
```
X_output_transf = modelo_cm["transf"].fit_transform(modelo_cm["encod"].transform(X_model))
```

In [177]:
```
prediction = modelo_cm["model"].predict_proba(X_output_transf)
```

In [178]:
```
strat = modelo_cm["strat"].predict(y_score=np.array(prediction[:,1]))
```

In [179]:
```
df['Score']= prediction[:,1]
df['Strat']= strat
```

In [180]:
```
print(df[['Strat']].value_counts())
```

```
Strat
Bajo        678299
Medio       446101
Alto        299332
Muy Alto    127378
dtype: int64
```

In [181]:
```
from IPython.display import Javascript
Javascript("Jupyter.notebook.execute_cell_range(1,8)")
```

Out[181]: