Proyecto de Fundamentos Analítica I *(Daniel Delgado Caicedo, Raúl Alberto Echeverry López, Luis Esteban Ordoñez Erazo, Fabian Salazar Figueroa)*

TABLA DE CONTENIDO

1. Introducción

Síndrome del ovario poliquístico (PCOS)

El síndrome de ovario poliquístico (PCOS) es un problema hormonal que afecta a las mujeres en la edad reproductiva. Aquí se presenta información relevante sobre el PCOS:

Síntomas y características del PCOS:

- **Períodos menstruales irregulares**: Las mujeres con PCOS pueden experimentar ciclos menstruales irregulares o ausencia de menstruación.
- **Exceso de vello corporal (hirsutismo)**: El PCOS puede causar un aumento en el vello facial, en el pecho, el abdomen y la espalda.
- **Acné**: Las alteraciones hormonales pueden provocar brotes de acné.
- **Calvicie**: Algunas mujeres con PCOS pueden experimentar adelgazamiento del cabello.
- **Dificultad para quedar embarazada**: El PCOS es una causa común de infertilidad.

Causas y diagnóstico: El PCOS está relacionado con un desequilibrio hormonal y problemas metabólicos. El diagnóstico se basa en la presencia de síntomas, análisis de sangre y ecografía para evaluar los ovarios.

Tratamiento: El tratamiento puede incluir cambios en el estilo de vida (dieta y ejercicio), medicamentos para regular los ciclos menstruales y mejorar la fertilidad, y en algunos casos, cirugía.

2. Contexto

Pregunta inteligente (SMART)

Que diferencias relevantes y patognomónicas existen entre las mujeres diagnosticadas con sindrome de ovario poliquistico y las que no presentan el síndrome en los 10 hospitales de Kerala, India?

- **Específico (Specific)**: Se busca saber si hay diferencias importantes entre las mujeres pacientes que se hicieron el dianostico para sindrome de ovario poliquistico en los 10 hospitales.
- **Medible (Measurable)**: A través de modelos estadísticos
- **Alcanzable (Achievable)**: La información y estudios previos permiten catalogarlo como alcanzable
- **Relevante (Relevant)**: Permitirá ayudar a comprender más y mejor el síndrome de ovario poliquístico.
- **Temporal (Time-bound)**: Se pretende estudiarlo durante 2 años

Objetivo

Evaluar si se evidencian diferencias patognomónicas de acuerdo a la información recopilada

Diccionario de Datos

- (s-n): Contador de pacientes
- paciente-id: Registro y asignación de id único y anónimo a cada paciente
- pcos(s-n): Resultado (Si = 1, No = 0) del examen de síndrome de ovario poliquístico de la paciente
- edad-a: Edad en años de la paciente al momento de diagnosticarse para pcos
- peso-kg: Peso en Kilogramos de la paciente al momento de diagnosticarse para pcos
- estatura-cm: Estatura en centimentro de la paciente al momento de diagnosticarse para pcos
- imc: Índice de masa corporal de la paciente al momento de diagnosticarse para pcos
- grupo-sanguineo: Factor sanguíneo RH de la paciente
- frecuencia-cardiaca-bpm: Frecuencia cardiaca medida en pulso por minuto de la paciente al momento de diagnosticarse para pcos
- frecuencia-respiratoria-respiraciones/min: Frecuencia respiratoria medida en respiraciones por minuto de la paciente al momento de diagnosticarse para pcos
- hemoglobina-g/dl: Hemoglobina de la paciente expresada en gramos por decilitro al momento de diagnosticarse para pcos
- ciclo-r/i: Duración del flujo menstrual de la paciente
- duracion-ciclo-d: Duración del ciclo mesntrual de la paciente medido en dias
- tiempo-casada-a: Tiempo de casada de la paciente medido en años al momento de diagnosticarse para pcos
- embarazada(s-n): Resultado de prueba de embarazo (Si = 1, No = 0) de la paciente al momento de diagnosticarse para pcos
- nro-abortos: Número de abortos que ha sufrido la paciente hasta el momento de diagnosticarse para pcos
- h-beta-hcg-I-mIU/mL: Resultado de la prueba de la hormona gonadotropina coriónica humana (hCG) de la paciente al momento de diagnosticarse para pcos
- h-beta-hcg-II-mIU/mL: Resultado de la segunda prueba de la hormona gonadotropina coriónica humana (hCG) de la paciente al momento de diagnosticarse para pcos
- h-fsh-mIU/mL: Resultado de la prueba de la hormona foliculoestimulante (FSH) de la paciente al momento de diagnosticarse para pcos
- h-lh-mIU/mL: Resultado de la prueba de la hormona luteinizante (LH) de la paciente al momento de diagnosticarse para pcos
- h-fsh/h-lh: Resultado del índice del cociente entre La hormona foliculoestimulante (FSH) y la hormona luteinizante (LH) de la paciente al momento de diagnosticarse pcos
- cadera-pulg: Medida de la cadera de la paciente en pulgadas al momento de diagnosticarse pcos
- cintura-pulg: Medida de la cintura de la paciente en pulgadas al momento de diagnosticarse pcos
- ind-cintura/cadera: Resultado del índice del cociente entre la medida de la cintura y la medida de la cadera de la paciente al momento de diagnosticarse pcos
- h-tsh-mIU/L: Resultado de la prueba de la hormona estimulante de la tiroides (TSH) de la paciente al momento de diagnosticarse pcos
- h-amh-ng/mL: Resultado de la prueba de la hormona antimülleriana (AMH) de la paciente al momento de diagnosticarse pcos

- h-prl-ng/mL: Resultado de la hormona prolactina (PRL) de la paciente al momento de diagnosticarse pcos
- ex-vit-d3-ng/mL: Resultado del examen de la vitamina D de la paciente al momento de diagnosticarse pcos
- h-prg-ng/mL: Resultado de la hormona progesterona (PRG) de la paciente al momento de diagnosticarse pcos
- ex-rbs-mg/dl: Resultado del examen del análisis de glucosa en sangre al azar (RBS) de la paciente al momento de diagnosticarse pcos
- ganancia-peso(s-n): Respuesta (Si = 1, No = 0) de la paciente comparando su peso al momento de diagnosticarse pcos contra el control médico inmediantamente anterior
- crecimiento-cabello(s-n): Respuesta (Si = 1, No = 0) de la paciente comparando su largo de cabello al momento de diagnosticarse pcos contra el control médico inmediantamente anterior
- oscurecimiento-piel(s-n): Respuesta (Si = 1, No = 0) de la paciente comparando si hay oscurecimeinto de su piel al momento de diagnosticarse pcos contra el control médico inmediantamente anterior
- perdida-cabello(s-n): Respuesta (Si = 1, No = 0) de la paciente comparando si hay pérdida de cabello al momento de diagnosticarse pcos contra el control médico inmediantamente anterior
- barro-espinilla(s-n): Respuesta (Si = 1, No = 0) de la paciente comparando si hay aumento o aparación de espinillas, barros o granos en la piel al momento de diagnosticarse pcos contra el control médico inmediantamente anterior
- comida-rapida(s-n): Respuesta (Si = 1, No = 0) de la paciente sobre su alimentación con comidas rapidas o "chatarras"
- ejercicio-regular(s-n): Respuesta (Si = 1, No = 0) de la paciente sobre su ejercitación regular
- ps-sistolica-mmHg: Resultado de la presión sanguinea sistólica de la paciente al momento de diagnosticarse pcos
- ps-diastolica-mmHg: Resultado de la presión sanguinea diastólica de la paciente al momento de diagnosticarse pcos
- nro-foliculos-ovario-izq: Número de folículos antrales en el ovario izquierdo de la paciente al momento de diagnosticarse pcos
- nro-foliculos-ovario-der: Número de folículos antrales en el ovario derecho de la paciente al momento de diagnosticarse pcos
- prom-tam-foliculos-ovario-izq-mm: Tamaño promedio de los folículos antrales en el ovario izquierdo de la paciente al momento de diagnosticarse pcos
- prom-tam-foliculos-ovario-der-mm: Tamaño promedio de los folículos antrales en el ovario derecho de la paciente al momento de diagnosticarse pcos
- endometrio-mm: Tamaño del endometrio de la paciente al momento de diagnosticarse pcos medido en milimetros

Convenciones

1. **pcos** significa síndrome de ovario poliquistico
2. Por cada pregunta de respuesta tipo si-no (si = 1, no = 0), se identifica en el encabezado de cada parametro al final del nombre del mismo así: (s-n)
3. Grupo sanguíneo se define así: (A+ = 11, A- = 12, B+ = 13, B-= 14, O+ = 15, O- = 16, AB+ = 17, AB- = 18)

4. Presión sanguínea ingresada como sistólica y diastólica de forma separada
5. Casos de la hormona Beta-HCG son mensionados como **h-beta-hcg-I-mIU/mL** y **h-beta-hcg-II-mIU/mL**
6. El prefijo **h** anterior al primer **"-"** en el nombre de cada parámetro significa hormona. Ejemplo: **h**-beta-hcg-I-mIU/mL, donde ese prefijo **h** significa hormona.
7. El prefijo **ex** anterior al primer **"-"** en el nombre de cada parámetro significa examen
8. Las unidades de medida estarán posteriores al último **"-"** en el nombre de cada parámetro y respetarán la nomenclatura médica. Ejemplo: h-beta-hcg-I-**mIU/mL**, donde su unidad de medida es: **mIU/mL**
9. Cada fila representa la información y resultados de cada paciente en el proceso de diagnosticarse para pcos

Dataset tomado de:

author = {Prasoon Kottarathil}, title = {Polycystic ovary syndrome (PCOS)}, year = {2020}, publisher = {kaggle}, journal = {Kaggle Dataset}, how published = {\url{https://www.kaggle.com/prasoonkottarathil/polycystic-ovary-syndrome-pcos}} (https://www.kaggle.com/prasoonkottarathil/polycystic-ovary-syndrome-pcos%7D%7D)

El **Dataset** contiene todos los **parámetros físicos y clínicos** para determinar problemas relacionados con el **síndrome de ovario poliquístico (PCOS)** y la **infertilidad**. Los datos se recopilaron en **10 hospitales diferentes en Kerala, India**.

3. Análisis exploratorio

Importación de bibliotecas

In [ ]:
```
# Importacion de pandas y asignándole el alias pd, se usa para el tratamiento
import pandas as pd
# Importacion de numpy y asignándole el alias np, se usa para operaciones matr
import numpy as np
# Importacion de matplotlib.pyplot y asignándole el alias plt, se usa para gra
import matplotlib.pyplot as plt
# Importacion de plotly.express y asignándole el alias px, se usa para grafica
import plotly.express as px
# Importacion de seaborn y asignándole el alias sns, se usa para graficar junt
import seaborn as sns
# Importacion de enable_iterative_imputer de la biblioteca scikit-learn (tambi
# se usa para poder importar iterativeimputer
from sklearn.experimental import enable_iterative_imputer
# Importacion de iterativeimputer de la biblioteca sklearn, se usa para imputa
from sklearn.impute import IterativeImputer
# importa la clase Pipeline de la biblioteca scikit-learn (también conocida co
# se usa para construir flujos de trabajo
from sklearn.pipeline import Pipeline
# importa la clase StandardScaler de la biblioteca scikit-learn (también conoc
# se usa para estandarizar las caracteristicas de los datos
from sklearn.preprocessing import StandardScaler
# importa la clase PCA de la biblioteca scikit-learn (también conocida como sk
# se usa para reducir dimension de los datos y encontrar caracteristicas más i
from sklearn.decomposition import PCA
# importa la clase LogisticRegression de la biblioteca scikit-learn (también c
# se usa para hacer una regresión logística, herramietna para clasificacion
from sklearn.linear_model import LogisticRegression
# importa la clase KMeans de la biblioteca scikit-learn (también conocida como
# se usa para hacer una imputacion por clustering
from sklearn.cluster import KMeans
# importa la clase SimpleImputer de la biblioteca scikit-learn (también conoci
# se usa para hacer una imputacion por clustering
from sklearn.impute import SimpleImputer
# importa la clase train_test_split de la biblioteca scikit-learn (también con
# se usa para hacer una modelacion
from sklearn.model_selection import train_test_split
# importa la clase RandomForestClassifier de la biblioteca scikit-learn (tambi
# se usa para hacer una modelacion
from sklearn.ensemble import RandomForestClassifier
# importa la clase accuracy_score de la biblioteca scikit-learn (también conoc
# se usa para hacer una modelacion
from sklearn.metrics import accuracy_score
# importa la clase cross_val_score de la biblioteca scikit-learn (también cono
# se usa para hacer una modelacion
from sklearn.model_selection import cross_val_score
# importa la clase GridSearchCV de la biblioteca scikit-learn (también conocid
# se usa para hacer una modelacion
from sklearn.model_selection import GridSearchCV
# importa la clase confusion_matrix de la biblioteca scikit-learn (también con
# se usa para hacer una modelacion
from sklearn.metrics import confusion_matrix
# importa la clase classification_report de la biblioteca scikit-learn (tambié
# se usa para hacer una modelacion
```

```
from sklearn.metrics import classification_report
```

Carga de la base de datos de los pacientes

In [ ]:
```
# Ruta al archivo CSV (local)
#ruta_archivo = r"C:\Users\alfa7\OneDrive\Documentos\PCOS_data.csv"

# Carga los datos desde el archivo CSV su separador de columnas es una coma (,
df = pd.read_csv('https://raw.githubusercontent.com/alfa7g7/Fundamentos-Analit
```

Exploración del conjunto de datos

In [ ]:
```
# Muestra en una tupla la cantidad de (filas, columnas) del DataFrame df
df.shape
```

Out[3]: (541, 44)

In [ ]:
```
# Muestra la información del DataFrame df, donde evidenciaremos el índice, las
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541 entries, 0 to 540
Data columns (total 44 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Sl. No                  541 non-null    int64  
 1   Patient File No.        541 non-null    int64  
 2   PCOS (Y/N)              541 non-null    int64  
 3    Age (yrs)              541 non-null    int64  
 4   Weight (Kg)             541 non-null    float64
 5   Height(Cm)              541 non-null    float64
 6   BMI                     541 non-null    float64
 7   Blood Group             541 non-null    int64  
 8   Pulse rate(bpm)         541 non-null    int64  
 9   RR (breaths/min)        541 non-null    int64  
 10  Hb(g/dl)                541 non-null    float64
 11  Cycle(R/I)              541 non-null    int64  
 12  Cycle length(days)      541 non-null    int64  
 13  Marraige Status (Yrs)   540 non-null    float64
 14  Pregnant(Y/N)           541 non-null    int64  
 15  No. of abortions        541 non-null    int64  
 16    I   beta-HCG(mIU/mL)  541 non-null    float64
 17  II    beta-HCG(mIU/mL)  541 non-null    object 
 18  FSH(mIU/mL)             541 non-null    float64
 19  LH(mIU/mL)              541 non-null    float64
 20  FSH/LH                  541 non-null    float64
 21  Hip(inch)               541 non-null    int64  
 22  Waist(inch)             541 non-null    int64  
 23  Waist:Hip Ratio         541 non-null    float64
 24  TSH (mIU/L)             541 non-null    float64
 25  AMH(ng/mL)              541 non-null    object 
 26  PRL(ng/mL)              541 non-null    float64
 27  Vit D3 (ng/mL)          541 non-null    float64
 28  PRG(ng/mL)              541 non-null    float64
 29  RBS(mg/dl)              541 non-null    float64
 30  Weight gain(Y/N)        541 non-null    int64  
 31  hair growth(Y/N)        541 non-null    int64  
 32  Skin darkening (Y/N)    541 non-null    int64  
 33  Hair loss(Y/N)          541 non-null    int64  
 34  Pimples(Y/N)            541 non-null    int64  
 35  Fast food (Y/N)         540 non-null    float64
 36  Reg.Exercise(Y/N)       541 non-null    int64  
 37  BP _Systolic (mmHg)     541 non-null    int64  
 38  BP _Diastolic (mmHg)    541 non-null    int64  
 39  Follicle No. (L)        541 non-null    int64  
 40  Follicle No. (R)        541 non-null    int64  
 41  Avg. F size (L) (mm)    541 non-null    float64
 42  Avg. F size (R) (mm)    541 non-null    float64
 43  Endometrium (mm)        541 non-null    float64
dtypes: float64(19), int64(23), object(2)
memory usage: 186.1+ KB
None
```

In [ ]:
```
# Observamos las primeras filas del dataframe
df.head()
```

Out[5]:

	Sl. No	Patient File No.	PCOS (Y/N)	Age (yrs)	Weight (Kg)	Height(Cm)	BMI	Blood Group	Pulse rate(bpm)	RR (breaths/min)	...	Pim
0	1	1	0	28	44.6	152.0	19.3	15	78	22	...	
1	2	2	0	36	65.0	161.5	24.9	15	74	20	...	
2	3	3	1	33	68.8	165.0	25.3	11	72	18	...	
3	4	4	0	37	65.0	148.0	29.7	13	72	20	...	
4	5	5	0	25	52.0	161.0	20.1	11	72	18	...	

5 rows × 44 columns

In [ ]:
```
#Observamos las últimas filas del dataframe
df.tail()
```

Out[6]:

	Sl. No	Patient File No.	PCOS (Y/N)	Age (yrs)	Weight (Kg)	Height(Cm)	BMI	Blood Group	Pulse rate(bpm)	RR (breaths/min)	...	F
536	537	537	0	35	50.0	164.592	18.5	17	72	16	...	
537	538	538	0	30	63.2	158.000	25.3	15	72	18	...	
538	539	539	0	36	54.0	152.000	23.4	13	74	20	...	
539	540	540	0	27	50.0	150.000	22.2	15	74	20	...	
540	541	541	1	23	82.0	165.000	30.1	13	80	20	...	

5 rows × 44 columns

In [ ]:
```
# Muestra el nombre de las columnas (llamdas comunmente headers) del DataFrame
print(df.columns)
```

```
Index(['Sl. No', 'Patient File No.', 'PCOS (Y/N)', ' Age (yrs)', 'Weight (K
g)',
       'Height(Cm) ', 'BMI', 'Blood Group', 'Pulse rate(bpm) ',
       'RR (breaths/min)', 'Hb(g/dl)', 'Cycle(R/I)', 'Cycle length(days)',
       'Marraige Status (Yrs)', 'Pregnant(Y/N)', 'No. of abortions',
       '  I   beta-HCG(mIU/mL)', 'II    beta-HCG(mIU/mL)', 'FSH(mIU/mL)',
       'LH(mIU/mL)', 'FSH/LH', 'Hip(inch)', 'Waist(inch)', 'Waist:Hip Ratio',
       'TSH (mIU/L)', 'AMH(ng/mL)', 'PRL(ng/mL)', 'Vit D3 (ng/mL)',
       'PRG(ng/mL)', 'RBS(mg/dl)', 'Weight gain(Y/N)', 'hair growth(Y/N)',
       'Skin darkening (Y/N)', 'Hair loss(Y/N)', 'Pimples(Y/N)',
       'Fast food (Y/N)', 'Reg.Exercise(Y/N)', 'BP _Systolic (mmHg)',
       'BP _Diastolic (mmHg)', 'Follicle No. (L)', 'Follicle No. (R)',
       'Avg. F size (L) (mm)', 'Avg. F size (R) (mm)', 'Endometrium (mm)'],
      dtype='object')
```

- **Hallazgos**
  - df.shape. Nos permitió saber que nuestro dataframe tiene 44 Columnas y 541 filas
  - print(df.info()). Nos indica que nuestro dataframe tiene:
    - 19 columnas con tipo de datos flotante
    - 23 columnas con tipo de dato entero
    - 2 columnas con tipo de datos objeto
    - Determinar que los headers o nombres de cada columnas no están homogéneos, presentan espacios, etc
    - Las variables 'Marraige Status (Yrs)' y 'Fast food (Y/N)' tienen 1 dato faltante o nulo cada una.
  - print(df.columns). Nos permite observar los headers y su nombre. (Esto debido a que debía analizarse más estos valores por lo observado con la anterior función (print(df.info())))

  Con esta información podemos determinar que se debe estandarizar las columnas o header y adicional procesar o trabajar estas columnas o variables categóricas para que podamos más adelante usar correctamente los modelos estadisticos.

Tipos de datos

- **Análisis de columnas o headers**

In [ ]:
```
# Definimos los nuevos nombres de las columnas o headers en un arreglo llamado
headers = ['(s-n)','paciente-id','pcos(s-n)','edad-a','peso-kg','estatura-cm',
           'frecuencia-cardiaca-bpm','frecuencia-respiratoria-respiraciones/mi
           'ciclo-r/i','duracion-ciclo-d','tiempo-casada-a','embarazada(s-n)',
           'h-beta-hcg-II-mIU/mL','h-fsh-mIU/mL','h-lh-mIU/mL','h-fsh/h-lh','c
           'ind-cintura/cadera','h-tsh-mIU/L','h-amh-ng/mL','h-prl-ng/mL','ex-
           'ex-rbs-mg/dl','ganancia-peso(s-n)','crecimiento-cabello(s-n)','osc
           'perdida-cabello(s-n)','barro-espinilla(s-n)','comida-rapida(s-n)',
           'ps-sistolica-mmHg','ps-diastolica-mmHg','nro-foliculos-ovario-izq'
           'prom-tam-foliculos-ovario-izq-mm','prom-tam-foliculos-ovario-der-m

# Asigno los nuevos nombres de encabezados o columnas a mi dataframe df
df.columns = headers

# Revisión impresa
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541 entries, 0 to 540
Data columns (total 44 columns):
 #   Column                                      Non-Null Count  Dtype  
---  ------                                      --------------  -----  
 0   (s-n)                                       541 non-null    int64  
 1   paciente-id                                 541 non-null    int64  
 2   pcos(s-n)                                   541 non-null    int64  
 3   edad-a                                      541 non-null    int64  
 4   peso-kg                                     541 non-null    float64
 5   estatura-cm                                 541 non-null    float64
 6   imc                                         541 non-null    float64
 7   grupo-sanguineo                             541 non-null    int64  
 8   frecuencia-cardiaca-bpm                     541 non-null    int64  
 9   frecuencia-respiratoria-respiraciones/min   541 non-null    int64  
 10  hemoglobina-g/dl                            541 non-null    float64
 11  ciclo-r/i                                   541 non-null    int64  
 12  duracion-ciclo-d                            541 non-null    int64  
 13  tiempo-casada-a                             540 non-null    float64
 14  embarazada(s-n)                             541 non-null    int64  
 15  nro-abortos                                 541 non-null    int64  
 16  h-beta-hcg-I-mIU/mL                         541 non-null    float64
 17  h-beta-hcg-II-mIU/mL                        541 non-null    object 
 18  h-fsh-mIU/mL                                541 non-null    float64
 19  h-lh-mIU/mL                                 541 non-null    float64
 20  h-fsh/h-lh                                  541 non-null    float64
 21  cadera-pulg                                 541 non-null    int64  
 22  cintura-pulg                                541 non-null    int64  
 23  ind-cintura/cadera                          541 non-null    float64
 24  h-tsh-mIU/L                                 541 non-null    float64
 25  h-amh-ng/mL                                 541 non-null    object 
 26  h-prl-ng/mL                                 541 non-null    float64
 27  ex-vit-d3-ng/mL                             541 non-null    float64
 28  h-prg-ng/mL                                 541 non-null    float64
 29  ex-rbs-mg/dl                                541 non-null    float64
 30  ganancia-peso(s-n)                          541 non-null    int64  
 31  crecimiento-cabello(s-n)                    541 non-null    int64  
 32  oscurecimiento-piel(s-n)                    541 non-null    int64  
 33  perdida-cabello(s-n)                        541 non-null    int64  
 34  barro-espinilla(s-n)                        541 non-null    int64  
 35  comida-rapida(s-n)                          540 non-null    float64
 36  ejercicio-regular(s-n)                      541 non-null    int64  
 37  ps-sistolica-mmHg                           541 non-null    int64  
 38  ps-diastolica-mmHg                          541 non-null    int64  
 39  nro-foliculos-ovario-izq                    541 non-null    int64  
 40  nro-foliculos-ovario-der                    541 non-null    int64  
 41  prom-tam-foliculos-ovario-izq-mm            541 non-null    float64
 42  prom-tam-foliculos-ovario-der-mm            541 non-null    float64
 43  endometrio-mm                               541 non-null    float64
dtypes: float64(19), int64(23), object(2)
memory usage: 186.1+ KB
None
```

- **Análisis variables categóricas**

En este caso particular solo nos interesan las dos columnas tipo objeto ya que las demás

In [ ]:
```
#Primera Columna tipo objeto
df["h-beta-hcg-II-mIU/mL"].head()
```

Out[9]:
```
0      1.99
1      1.99
2    494.08
3      1.99
4    801.45
Name: h-beta-hcg-II-mIU/mL, dtype: object
```

In [ ]:
```
#Segunda columna tipo objeto
df["h-amh-ng/mL"].head()
```

Out[10]:
```
0    2.07
1    1.53
2    6.63
3    1.22
4    2.26
Name: h-amh-ng/mL, dtype: object
```

In [ ]:
```
#Convertiremos los valores categoricos en numericos, ya que han sido numeros g
#Si hay algun valor que no se puede convertir a numerico se convierte a NaN (N
df["h-beta-hcg-II-mIU/mL"] = pd.to_numeric(df["h-beta-hcg-II-mIU/mL"], errors=
df["h-amh-ng/mL"] = pd.to_numeric(df["h-amh-ng/mL"], errors='coerce')
```

In [ ]:
```
# verificacion de la conversion
df["h-beta-hcg-II-mIU/mL"].head()
```

Out[12]:
```
0      1.99
1      1.99
2    494.08
3      1.99
4    801.45
Name: h-beta-hcg-II-mIU/mL, dtype: float64
```

In [ ]:
```
# verificacion de la conversion
df["h-amh-ng/mL"].head()
```

Out[13]:
```
0    2.07
1    1.53
2    6.63
3    1.22
4    2.26
Name: h-amh-ng/mL, dtype: float64
```

In [ ]:
```
# Revisión impresa, sin variables tipo objeto
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541 entries, 0 to 540
Data columns (total 44 columns):
 #   Column                                      Non-Null Count  Dtype  
---  ------                                      --------------  -----  
 0   (s-n)                                       541 non-null    int64  
 1   paciente-id                                 541 non-null    int64  
 2   pcos(s-n)                                   541 non-null    int64  
 3   edad-a                                      541 non-null    int64  
 4   peso-kg                                     541 non-null    float64
 5   estatura-cm                                 541 non-null    float64
 6   imc                                         541 non-null    float64
 7   grupo-sanguineo                             541 non-null    int64  
 8   frecuencia-cardiaca-bpm                     541 non-null    int64  
 9   frecuencia-respiratoria-respiraciones/min   541 non-null    int64  
 10  hemoglobina-g/dl                            541 non-null    float64
 11  ciclo-r/i                                   541 non-null    int64  
 12  duracion-ciclo-d                            541 non-null    int64  
 13  tiempo-casada-a                             540 non-null    float64
 14  embarazada(s-n)                             541 non-null    int64  
 15  nro-abortos                                 541 non-null    int64  
 16  h-beta-hcg-I-mIU/mL                         541 non-null    float64
 17  h-beta-hcg-II-mIU/mL                        540 non-null    float64
 18  h-fsh-mIU/mL                                541 non-null    float64
 19  h-lh-mIU/mL                                 541 non-null    float64
 20  h-fsh/h-lh                                  541 non-null    float64
 21  cadera-pulg                                 541 non-null    int64  
 22  cintura-pulg                                541 non-null    int64  
 23  ind-cintura/cadera                          541 non-null    float64
 24  h-tsh-mIU/L                                 541 non-null    float64
 25  h-amh-ng/mL                                 540 non-null    float64
 26  h-prl-ng/mL                                 541 non-null    float64
 27  ex-vit-d3-ng/mL                             541 non-null    float64
 28  h-prg-ng/mL                                 541 non-null    float64
 29  ex-rbs-mg/dl                                541 non-null    float64
 30  ganancia-peso(s-n)                          541 non-null    int64  
 31  crecimiento-cabello(s-n)                    541 non-null    int64  
 32  oscurecimiento-piel(s-n)                    541 non-null    int64  
 33  perdida-cabello(s-n)                        541 non-null    int64  
 34  barro-espinilla(s-n)                        541 non-null    int64  
 35  comida-rapida(s-n)                          540 non-null    float64
 36  ejercicio-regular(s-n)                      541 non-null    int64  
 37  ps-sistolica-mmHg                           541 non-null    int64  
 38  ps-diastolica-mmHg                          541 non-null    int64  
 39  nro-foliculos-ovario-izq                    541 non-null    int64  
 40  nro-foliculos-ovario-der                    541 non-null    int64  
 41  prom-tam-foliculos-ovario-izq-mm            541 non-null    float64
 42  prom-tam-foliculos-ovario-der-mm            541 non-null    float64
 43  endometrio-mm                               541 non-null    float64
dtypes: float64(21), int64(23)
memory usage: 186.1 KB
None
```

- **Análisis de datos faltantes o nulos**

In [ ]:
```
#Validacion de valores nulos en el dataframe muestra True quiere decir que hay
t = df.isnull().any().any()
print(t)
```

```
True
```

In [ ]:
```
#verificando valores nulos en los parametros o columnas del dataframe si muest
#tiene valores nulos
r = df.isnull().any()
print(r)
```

```
(s-n)                                          False
paciente-id                                    False
pcos(s-n)                                      False
edad-a                                         False
peso-kg                                        False
estatura-cm                                    False
imc                                            False
grupo-sanguineo                                False
frecuencia-cardiaca-bpm                        False
frecuencia-respiratoria-respiraciones/min      False
hemoglobina-g/dl                               False
ciclo-r/i                                      False
duracion-ciclo-d                               False
tiempo-casada-a                                 True
embarazada(s-n)                                False
nro-abortos                                    False
h-beta-hcg-I-mIU/mL                            False
h-beta-hcg-II-mIU/mL                            True
h-fsh-mIU/mL                                   False
h-lh-mIU/mL                                    False
h-fsh/h-lh                                     False
cadera-pulg                                    False
cintura-pulg                                   False
ind-cintura/cadera                             False
h-tsh-mIU/L                                    False
h-amh-ng/mL                                     True
h-prl-ng/mL                                    False
ex-vit-d3-ng/mL                                False
h-prg-ng/mL                                    False
ex-rbs-mg/dl                                   False
ganancia-peso(s-n)                             False
crecimiento-cabello(s-n)                       False
oscurecimiento-piel(s-n)                       False
perdida-cabello(s-n)                           False
barro-espinilla(s-n)                           False
comida-rapida(s-n)                              True
ejercicio-regular(s-n)                         False
ps-sistolica-mmHg                              False
ps-diastolica-mmHg                             False
nro-foliculos-ovario-izq                       False
nro-foliculos-ovario-der                       False
prom-tam-foliculos-ovario-izq-mm               False
prom-tam-foliculos-ovario-der-mm               False
endometrio-mm                                  False
dtype: bool
```

Identificamos las siguientes columnas o parámetros con valores nulos:

- tiempo-casada-a
- h-beta-hcg-II-mIU/mL

- h-amh-ng/mL
- comida-rapida(s-n)

IMPUTACION DE VALORES FALTANTES O NULOS

La imputación de valores faltantes dio lugar a una comparación entre 2 tipos de imputación al final nos decidimos por la imputación múltiple por ecuaciones encadenadas (MICE) debido a que:

- El algoritmo MICE asume que los datos faltan al azar (MAR), es decir, la falta de un campo se puede explicar por los valores de otras columnas, pero no de esa columna

MICE

```
Verificación de valores faltantes y llenado de datos faltantes o nulo
s en los parámetros o columnas del dataframe con imputacion MICE
```

In [ ]:
```
# Imputacion de datos por MICE
# Inicializa el imputador MICE
mice_imputer = IterativeImputer(max_iter=10, random_state=0)
# Realiza la imputación
df_imputed = mice_imputer.fit_transform(df)
# El resultado es un array NumPy, así que puedes convertirlo de nuevo a DataFr
df_imputed = pd.DataFrame(df_imputed, columns=df.columns)
#df = df_imputed
print(df_imputed.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541 entries, 0 to 540
Data columns (total 44 columns):
 #   Column                                    Non-Null Count  Dtype  
---  ------                                    --------------  -----  
 0   (s-n)                                     541 non-null    float64
 1   paciente-id                               541 non-null    float64
 2   pcos(s-n)                                 541 non-null    float64
 3   edad-a                                    541 non-null    float64
 4   peso-kg                                   541 non-null    float64
 5   estatura-cm                               541 non-null    float64
 6   imc                                       541 non-null    float64
 7   grupo-sanguineo                           541 non-null    float64
 8   frecuencia-cardiaca-bpm                   541 non-null    float64
 9   frecuencia-respiratoria-respiraciones/min 541 non-null    float64
 10  hemoglobina-g/dl                          541 non-null    float64
 11  ciclo-r/i                                 541 non-null    float64
 12  duracion-ciclo-d                          541 non-null    float64
 13  tiempo-casada-a                           541 non-null    float64
 14  embarazada(s-n)                           541 non-null    float64
 15  nro-abortos                               541 non-null    float64
 16  h-beta-hcg-I-mIU/mL                       541 non-null    float64
 17  h-beta-hcg-II-mIU/mL                      541 non-null    float64
 18  h-fsh-mIU/mL                              541 non-null    float64
 19  h-lh-mIU/mL                               541 non-null    float64
 20  h-fsh/h-lh                                541 non-null    float64
 21  cadera-pulg                               541 non-null    float64
 22  cintura-pulg                              541 non-null    float64
 23  ind-cintura/cadera                        541 non-null    float64
 24  h-tsh-mIU/L                               541 non-null    float64
 25  h-amh-ng/mL                               541 non-null    float64
 26  h-prl-ng/mL                               541 non-null    float64
 27  ex-vit-d3-ng/mL                           541 non-null    float64
 28  h-prg-ng/mL                               541 non-null    float64
 29  ex-rbs-mg/dl                              541 non-null    float64
 30  ganancia-peso(s-n)                        541 non-null    float64
 31  crecimiento-cabello(s-n)                  541 non-null    float64
 32  oscurecimiento-piel(s-n)                  541 non-null    float64
 33  perdida-cabello(s-n)                      541 non-null    float64
 34  barro-espinilla(s-n)                      541 non-null    float64
 35  comida-rapida(s-n)                        541 non-null    float64
 36  ejercicio-regular(s-n)                    541 non-null    float64
 37  ps-sistolica-mmHg                         541 non-null    float64
 38  ps-diastolica-mmHg                        541 non-null    float64
 39  nro-foliculos-ovario-izq                  541 non-null    float64
 40  nro-foliculos-ovario-der                  541 non-null    float64
 41  prom-tam-foliculos-ovario-izq-mm          541 non-null    float64
 42  prom-tam-foliculos-ovario-der-mm          541 non-null    float64
 43  endometrio-mm                             541 non-null    float64
dtypes: float64(44)
memory usage: 186.1 KB
None
```

In [ ]:
```
# Cambio las variables con datos ya imputados solamente al dataframe original
df['tiempo-casada-a'] = df_imputed['tiempo-casada-a']
df['h-beta-hcg-II-mIU/mL'] = df_imputed['h-beta-hcg-II-mIU/mL']
df['h-amh-ng/mL'] = df_imputed['h-amh-ng/mL']
# redondeamos la siguiente variable imputada para evitarun error en su respues
# (Corrección hecha después de evaluación gráfica inicial)
df_imputed['comida-rapida(s-n)'] = df_imputed['comida-rapida(s-n)'].round()
df['comida-rapida(s-n)'] = df_imputed['comida-rapida(s-n)']
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541 entries, 0 to 540
Data columns (total 44 columns):
 #   Column                                      Non-Null Count  Dtype  
---  ------                                      --------------  -----  
 0   (s-n)                                       541 non-null    int64  
 1   paciente-id                                 541 non-null    int64  
 2   pcos(s-n)                                   541 non-null    int64  
 3   edad-a                                      541 non-null    int64  
 4   peso-kg                                     541 non-null    float64
 5   estatura-cm                                 541 non-null    float64
 6   imc                                         541 non-null    float64
 7   grupo-sanguineo                             541 non-null    int64  
 8   frecuencia-cardiaca-bpm                     541 non-null    int64  
 9   frecuencia-respiratoria-respiraciones/min   541 non-null    int64  
 10  hemoglobina-g/dl                            541 non-null    float64
 11  ciclo-r/i                                   541 non-null    int64  
 12  duracion-ciclo-d                            541 non-null    int64  
 13  tiempo-casada-a                             541 non-null    float64
 14  embarazada(s-n)                             541 non-null    int64  
 15  nro-abortos                                 541 non-null    int64  
 16  h-beta-hcg-I-mIU/mL                         541 non-null    float64
 17  h-beta-hcg-II-mIU/mL                        541 non-null    float64
 18  h-fsh-mIU/mL                                541 non-null    float64
 19  h-lh-mIU/mL                                 541 non-null    float64
 20  h-fsh/h-lh                                  541 non-null    float64
 21  cadera-pulg                                 541 non-null    int64  
 22  cintura-pulg                                541 non-null    int64  
 23  ind-cintura/cadera                          541 non-null    float64
 24  h-tsh-mIU/L                                 541 non-null    float64
 25  h-amh-ng/mL                                 541 non-null    float64
 26  h-prl-ng/mL                                 541 non-null    float64
 27  ex-vit-d3-ng/mL                             541 non-null    float64
 28  h-prg-ng/mL                                 541 non-null    float64
 29  ex-rbs-mg/dl                                541 non-null    float64
 30  ganancia-peso(s-n)                          541 non-null    int64  
 31  crecimiento-cabello(s-n)                    541 non-null    int64  
 32  oscurecimiento-piel(s-n)                    541 non-null    int64  
 33  perdida-cabello(s-n)                        541 non-null    int64  
 34  barro-espinilla(s-n)                        541 non-null    int64  
 35  comida-rapida(s-n)                          541 non-null    float64
 36  ejercicio-regular(s-n)                      541 non-null    int64  
 37  ps-sistolica-mmHg                           541 non-null    int64  
 38  ps-diastolica-mmHg                          541 non-null    int64  
 39  nro-foliculos-ovario-izq                    541 non-null    int64  
 40  nro-foliculos-ovario-der                    541 non-null    int64  
 41  prom-tam-foliculos-ovario-izq-mm            541 non-null    float64
 42  prom-tam-foliculos-ovario-der-mm            541 non-null    float64
 43  endometrio-mm                               541 non-null    float64
dtypes: float64(21), int64(23)
memory usage: 186.1 KB
```

In [ ]:
```
df['(s-n)'].nunique()
```

Out[20]:
```
541
```

In [ ]:
```
df['paciente-id'].nunique()
```

Out[21]:
```
541
```

Observando la estructura del dataframe procedemos a sacar esas dos variables ya que ambas son identificadores tipo contador para enamurar cada uno de los pacientes del estudio, probablemente usadas incialmente para encriptar o enmascarar los datos sensibles de cada paciente (Nombres o identificacion).

In [ ]:
```
df.drop(columns = ['(s-n)', 'paciente-id'], axis =1, inplace = True)
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541 entries, 0 to 540
Data columns (total 42 columns):
 #   Column                                        Non-Null Count  Dtype  
---  ------                                        --------------  -----  
 0   pcos(s-n)                                     541 non-null    int64  
 1   edad-a                                        541 non-null    int64  
 2   peso-kg                                       541 non-null    float64
 3   estatura-cm                                   541 non-null    float64
 4   imc                                           541 non-null    float64
 5   grupo-sanguineo                               541 non-null    int64  
 6   frecuencia-cardiaca-bpm                       541 non-null    int64  
 7   frecuencia-respiratoria-respiraciones/min     541 non-null    int64  
 8   hemoglobina-g/dl                              541 non-null    float64
 9   ciclo-r/i                                     541 non-null    int64  
 10  duracion-ciclo-d                              541 non-null    int64  
 11  tiempo-casada-a                               541 non-null    float64
 12  embarazada(s-n)                               541 non-null    int64  
 13  nro-abortos                                   541 non-null    int64  
 14  h-beta-hcg-I-mIU/mL                           541 non-null    float64
 15  h-beta-hcg-II-mIU/mL                          541 non-null    float64
 16  h-fsh-mIU/mL                                  541 non-null    float64
 17  h-lh-mIU/mL                                   541 non-null    float64
 18  h-fsh/h-lh                                    541 non-null    float64
 19  cadera-pulg                                   541 non-null    int64  
 20  cintura-pulg                                  541 non-null    int64  
 21  ind-cintura/cadera                            541 non-null    float64
 22  h-tsh-mIU/L                                   541 non-null    float64
 23  h-amh-ng/mL                                   541 non-null    float64
 24  h-prl-ng/mL                                   541 non-null    float64
 25  ex-vit-d3-ng/mL                               541 non-null    float64
 26  h-prg-ng/mL                                   541 non-null    float64
 27  ex-rbs-mg/dl                                  541 non-null    float64
 28  ganancia-peso(s-n)                            541 non-null    int64  
 29  crecimiento-cabello(s-n)                      541 non-null    int64  
 30  oscurecimiento-piel(s-n)                      541 non-null    int64  
 31  perdida-cabello(s-n)                          541 non-null    int64  
 32  barro-espinilla(s-n)                          541 non-null    int64  
 33  comida-rapida(s-n)                            541 non-null    float64
 34  ejercicio-regular(s-n)                        541 non-null    int64  
 35  ps-sistolica-mmHg                             541 non-null    int64  
 36  ps-diastolica-mmHg                            541 non-null    int64  
 37  nro-foliculos-ovario-izq                      541 non-null    int64  
 38  nro-foliculos-ovario-der                      541 non-null    int64  
 39  prom-tam-foliculos-ovario-izq-mm              541 non-null    float64
 40  prom-tam-foliculos-ovario-der-mm              541 non-null    float64
 41  endometrio-mm                                 541 non-null    float64
dtypes: float64(21), int64(21)
memory usage: 177.6 KB
```

In [ ]:
```
#verificando valores nulos en los parametros o columnas del dataframe después
r = df.isnull().any()
print(r)
```

```
pcos(s-n)                                    False
edad-a                                       False
peso-kg                                      False
estatura-cm                                  False
imc                                          False
grupo-sanguineo                              False
frecuencia-cardiaca-bpm                      False
frecuencia-respiratoria-respiraciones/min    False
hemoglobina-g/dl                             False
ciclo-r/i                                    False
duracion-ciclo-d                             False
tiempo-casada-a                              False
embarazada(s-n)                              False
nro-abortos                                  False
h-beta-hcg-I-mIU/mL                          False
h-beta-hcg-II-mIU/mL                         False
h-fsh-mIU/mL                                 False
h-lh-mIU/mL                                  False
h-fsh/h-lh                                   False
cadera-pulg                                  False
cintura-pulg                                 False
ind-cintura/cadera                           False
h-tsh-mIU/L                                  False
h-amh-ng/mL                                  False
h-prl-ng/mL                                  False
ex-vit-d3-ng/mL                              False
h-prg-ng/mL                                  False
ex-rbs-mg/dl                                 False
ganancia-peso(s-n)                           False
crecimiento-cabello(s-n)                     False
oscurecimiento-piel(s-n)                     False
perdida-cabello(s-n)                         False
barro-espinilla(s-n)                         False
comida-rapida(s-n)                           False
ejercicio-regular(s-n)                       False
ps-sistolica-mmHg                            False
ps-diastolica-mmHg                           False
nro-foliculos-ovario-izq                     False
nro-foliculos-ovario-der                     False
prom-tam-foliculos-ovario-izq-mm             False
prom-tam-foliculos-ovario-der-mm             False
endometrio-mm                                False
dtype: bool
```

In [ ]:
```
#Validacion de que no queden valores nulos en el dataframe Si muestra False qu
t = df.isnull().any().any()
print(t)
```

```
False
```

Descripción estadística

In [ ]:
```
# Resumen estadístico del dataframe df
df.describe()
```

Out[26]:

	pcos(s-n)	edad-a	peso-kg	estatura-cm	imc	grupo-sanguineo	frecuencia-cardiaca-bpm	resp
count	541.000000	541.000000	541.000000	541.000000	541.000000	541.000000	541.000000	
mean	0.327172	31.430684	59.637153	156.484835	24.307579	13.802218	73.247689	
std	0.469615	5.411006	11.028287	6.033545	4.055129	1.840812	4.430285	
min	0.000000	20.000000	31.000000	137.000000	12.400000	11.000000	13.000000	
25%	0.000000	28.000000	52.000000	152.000000	21.600000	13.000000	72.000000	
50%	0.000000	31.000000	59.000000	156.000000	24.200000	14.000000	72.000000	
75%	1.000000	35.000000	65.000000	160.000000	26.600000	15.000000	74.000000	
max	1.000000	48.000000	108.000000	180.000000	38.900000	18.000000	82.000000	

8 rows × 42 columns

- **Análisis de outliers**

In [ ]:
```
df.max()
```

Out[27]:
```
pcos(s-n)                                    1.00
edad-a                                      48.00
peso-kg                                    108.00
estatura-cm                                180.00
imc                                         38.90
grupo-sanguineo                             18.00
frecuencia-cardiaca-bpm                     82.00
frecuencia-respiratoria-respiraciones/min   28.00
hemoglobina-g/dl                            14.80
ciclo-r/i                                    5.00
duracion-ciclo-d                            12.00
tiempo-casada-a                             30.00
embarazada(s-n)                              1.00
nro-abortos                                  5.00
h-beta-hcg-I-mIU/mL                      32460.97
h-beta-hcg-II-mIU/mL                     25000.00
h-fsh-mIU/mL                              5052.00
h-lh-mIU/mL                               2018.00
h-fsh/h-lh                                1372.83
cadera-pulg                                 48.00
cintura-pulg                                47.00
ind-cintura/cadera                           0.98
h-tsh-mIU/L                                 65.00
h-amh-ng/mL                                 66.00
h-prl-ng/mL                                128.24
ex-vit-d3-ng/mL                           6014.66
h-prg-ng/mL                                 85.00
ex-rbs-mg/dl                               350.00
ganancia-peso(s-n)                           1.00
crecimiento-cabello(s-n)                     1.00
oscurecimiento-piel(s-n)                     1.00
perdida-cabello(s-n)                         1.00
barro-espinilla(s-n)                         1.00
comida-rapida(s-n)                           1.00
ejercicio-regular(s-n)                       1.00
ps-sistolica-mmHg                          140.00
ps-diastolica-mmHg                         100.00
nro-foliculos-ovario-izq                    22.00
nro-foliculos-ovario-der                    20.00
prom-tam-foliculos-ovario-izq-mm            24.00
prom-tam-foliculos-ovario-der-mm            24.00
endometrio-mm                               18.00
dtype: float64
```

In [ ]:
```
# Exportacion del dataframe para creacion de modelados
#df.to_csv("data.csv")
```

In [ ]:
```
px.box(df)
```

Análisis gráfico variables categóricas

In [ ]:
```
# Gráfico para las variables categoricas SI = 1 - NO = 0
px.box(df, y = ['pcos(s-n)', 'embarazada(s-n)', 'ganancia-peso(s-n)', 'crecimi
                'oscurecimiento-piel(s-n)', 'perdida-cabello(s-n)', 'barro-esp
                'comida-rapida(s-n)', 'ejercicio-regular(s-n)'])
```

In [ ]:
```
px.histogram(df['pcos(s-n)'])
```

In [ ]:
```
px.histogram(df['embarazada(s-n)'])
```

In [ ]:
```
px.histogram(df['crecimiento-cabello(s-n)'])
```

In [ ]:
```
px.histogram(df['oscurecimiento-piel(s-n)'])
```

In [ ]:
```
px.histogram(df['perdida-cabello(s-n)'])
```

In [ ]:
```
px.histogram(df['ganancia-peso(s-n)'])
```

In [ ]:
```
px.histogram(df['barro-espinilla(s-n)'])
```

In [ ]:
```
px.histogram(df['comida-rapida(s-n)'])
```

In [ ]:
```
px.histogram(df['ejercicio-regular(s-n)'])
```

Análisis gráfico variables no categóricas

In [ ]:
```
px.box(df['edad-a'], points='all')
```

In [ ]:
```
px.box(df['peso-kg'], points='all')
```

In [ ]:
```
px.box(df['estatura-cm'], points='all')
```

In [ ]:
```
px.box(df['imc'], points='all')
```

In [ ]:
```
px.box(df['grupo-sanguineo'], points='all')
```

In [ ]:
```
px.box(df['frecuencia-cardiaca-bpm'], points='all')
```

In [ ]:
```
px.box(df['frecuencia-respiratoria-respiraciones/min'], points='all')
```

In [ ]:
```
px.box(df['hemoglobina-g/dl'], points='all')
```

In [ ]:
```
px.box(df['ciclo-r/i'], points='all')
```

In [ ]:
```
px.box(df['duracion-ciclo-d'], points='all')
```

In [ ]:
```
px.box(df['tiempo-casada-a'], points='all')
```

In [ ]:
```
px.box(df['nro-abortos'], points='all')
```

In [ ]:
```
px.histogram(df['nro-abortos'])
```

In [ ]:
```
px.box(df['h-beta-hcg-I-mIU/mL'], points='all')
```

In [ ]:
```
px.histogram(df['h-beta-hcg-I-mIU/mL'])
```

In [ ]:
```
px.box(df['h-beta-hcg-II-mIU/mL'], points='all')
```

In [ ]:
```
px.histogram(df['h-beta-hcg-II-mIU/mL'])
```

In [ ]:
```
px.box(df['h-fsh-mIU/mL'], points='all')
```

In [ ]:
```
px.histogram(df['h-fsh-mIU/mL'])
```

In [ ]:
```
px.box(df['h-lh-mIU/mL'], points='all')
```

In [ ]:
```
px.box(df['h-fsh/h-lh'], points='all')
```

In [ ]:
```
px.box(df['cadera-pulg'], points='all')
```

In [ ]:
```
px.box(df['cintura-pulg'], points='all')
```

In [ ]:
```
px.box(df['ind-cintura/cadera'], points='all')
```

In [ ]:
```
px.box(df['h-tsh-mIU/L'], points='all')
```

In [ ]:
```
px.box(df['h-amh-ng/mL'], points='all')
```

In [ ]:
```
px.box(df['h-prl-ng/mL'], points='all')
```

In [ ]:
```
px.box(df['ex-vit-d3-ng/mL'], points='all')
```

In [ ]:
```
px.box(df['h-prg-ng/mL'], points='all')
```

In [ ]:
```
px.box(df['ex-rbs-mg/dl'], points='all')
```

In [ ]:
```
px.box(df['ps-sistolica-mmHg'], points='all')
```

In [ ]:
```
px.box(df['ps-diastolica-mmHg'], points='all')
```

In [ ]:
```
px.box(df['nro-foliculos-ovario-izq'], points='all')
```

In [ ]:
```
px.box(df['nro-foliculos-ovario-der'], points='all')
```

In [ ]:
```
px.box(df['prom-tam-foliculos-ovario-izq-mm'], points='all')
```

In [ ]:
```
px.box(df['prom-tam-foliculos-ovario-der-mm'], points='all')
```

In [ ]:
```
px.box(df['endometrio-mm'], points='all')
```

In [ ]:
```
#Histograma de variable determinada para remover outliers
figura1 = px.histogram(df, x='ps-diastolica-mmHg', nbins=30)
figura1.show()
```

In [ ]:
```
#Histograma de variable determinada para remover outliers
figura2 = px.histogram(df, x='ps-sistolica-mmHg', nbins=30)
figura2.show()
```

In [ ]:
```
#Histograma de variable determinada para remover outliers
figura3 = px.histogram(df, x='endometrio-mm', nbins=30)
figura3.show()
```

In [ ]:
```
#Histograma de variable determinada para remover outliers
figura4 = px.histogram(df, x='prom-tam-foliculos-ovario-izq-mm', nbins=60)
figura4.show()
```

In [ ]:
```
#Histograma de variable determinada para remover outliers
figura5 = px.histogram(df, x='prom-tam-foliculos-ovario-der-mm', nbins=60)
figura5.show()
```

In [ ]:
```
#Histograma de variable determinada para remover outliers
figura6 = px.histogram(df, x='frecuencia-cardiaca-bpm', nbins=30)
figura6.show()
```

In [ ]:
```
# Teniendo en cuenta el concepto medico procedemos a identificar y sacar datos
# Aquí discrepamos y preferimos mantener ciertos valores que resultan mas real
# a largo plazo en algunos outliers propuestos por la autora original del estu

df = df[(df['ps-diastolica-mmHg']>20)]
df = df[(df['ps-sistolica-mmHg']>20)]
df = df[(df['endometrio-mm']>0)]
df = df[(df['prom-tam-foliculos-ovario-izq-mm']>0)]
df = df[(df['prom-tam-foliculos-ovario-der-mm']>0)]
df = df[(df['frecuencia-cardiaca-bpm']>20)]
df = df[(df['ciclo-r/i']<4.5)]

df.shape
```

Out[83]: (526, 42)

Los motivos de exclusion para estos registros fueron tomados teniendo en cuenta el concepto de los expecialistas en el área específica de conocimiento. Según concepto de ellos:

- las presiones diastólicas, sistólicas y la frecuencia cardiaca inferiores en valor a 20 con incompatibles con la vida.
- Un endometrio con un tamaño menor a cero es un error de digitación por parte del personal encargado de generear estos informes inicialmente, teniendo en cuenta que estos alimentaron el dataframe para el estudio.
- El tamaño de los folículos en ambos ovarios si y solo sí, debe ser un valor positivo; en cualquier escenario contrario esto significa en error de llenado por parte del personal humano a la hora de rendir el informe con el cual se llenó el dataframe.
- En cuestion a los ciclos se determinó que era un varialble dicotómica por ende cualquier respues que estuviera fuera de su rango es decir un valor mayor a 4, significa o da a entender nuevamente un error de digitación por ende esos mismos son excluidos.

Estos criterios de exclusión fueron tenidos en cuenta porteriormentes a una reunión con especialistas en el dominio del tema

Como se muestra a continuación ya el valor mínimo es superior a cero en todos los parámetros excepto los que originalmente eran categóricos con respuesta (s-n), para los cuales no aplica el criterio.

In [ ]:
```
df.describe()
```

Out[84]:

	pcos(s-n)	edad-a	peso-kg	estatura-cm	imc	grupo-sanguineo	frecuencia-cardiaca-bpm	resp
count	526.000000	526.000000	526.000000	526.000000	526.000000	526.000000	526.000000	
mean	0.334601	31.418251	59.675475	156.381749	24.356464	13.809886	73.461977	
std	0.472300	5.417781	10.950673	5.989715	4.034686	1.843903	2.697703	
min	0.000000	20.000000	31.000000	137.000000	12.400000	11.000000	70.000000	
25%	0.000000	27.000000	52.000000	152.000000	21.900000	13.000000	72.000000	
50%	0.000000	31.000000	59.800000	156.000000	24.300000	14.500000	72.000000	
75%	1.000000	35.000000	65.000000	160.000000	26.700000	15.000000	74.000000	
max	1.000000	48.000000	108.000000	180.000000	38.900000	18.000000	82.000000	

8 rows × 42 columns

In [ ]:
```
px.box(df['ps-diastolica-mmHg'], points='all')
```

In [ ]:
```
#Histograma de variable sin el outlier
figura7 = px.histogram(df, x='ps-diastolica-mmHg', nbins=30)
figura7.show()
```

In [ ]:
```
px.box(df['ps-sistolica-mmHg'], points='all')
```

In [ ]:
```
#Histograma de variable sin outliers
figura8 = px.histogram(df, x='ps-sistolica-mmHg', nbins=30)
figura8.show()
```

In [ ]:
```
#Histograma de variable sin outliers
figura9 = px.histogram(df, x='endometrio-mm', nbins=30)
figura9.show()
```

In [ ]:
```
#Histograma de variable sin outliers
figura10 = px.histogram(df, x='prom-tam-foliculos-ovario-izq-mm', nbins=60)
figura10.show()
```

In [ ]:
```
#Histograma de variable sin outliers
figura11 = px.histogram(df, x='prom-tam-foliculos-ovario-der-mm', nbins=60)
figura11.show()
```

In [ ]:
```
#Histograma de variable sin outliers
figura12 = px.histogram(df, x='frecuencia-cardiaca-bpm', nbins=30)
figura12.show()
```

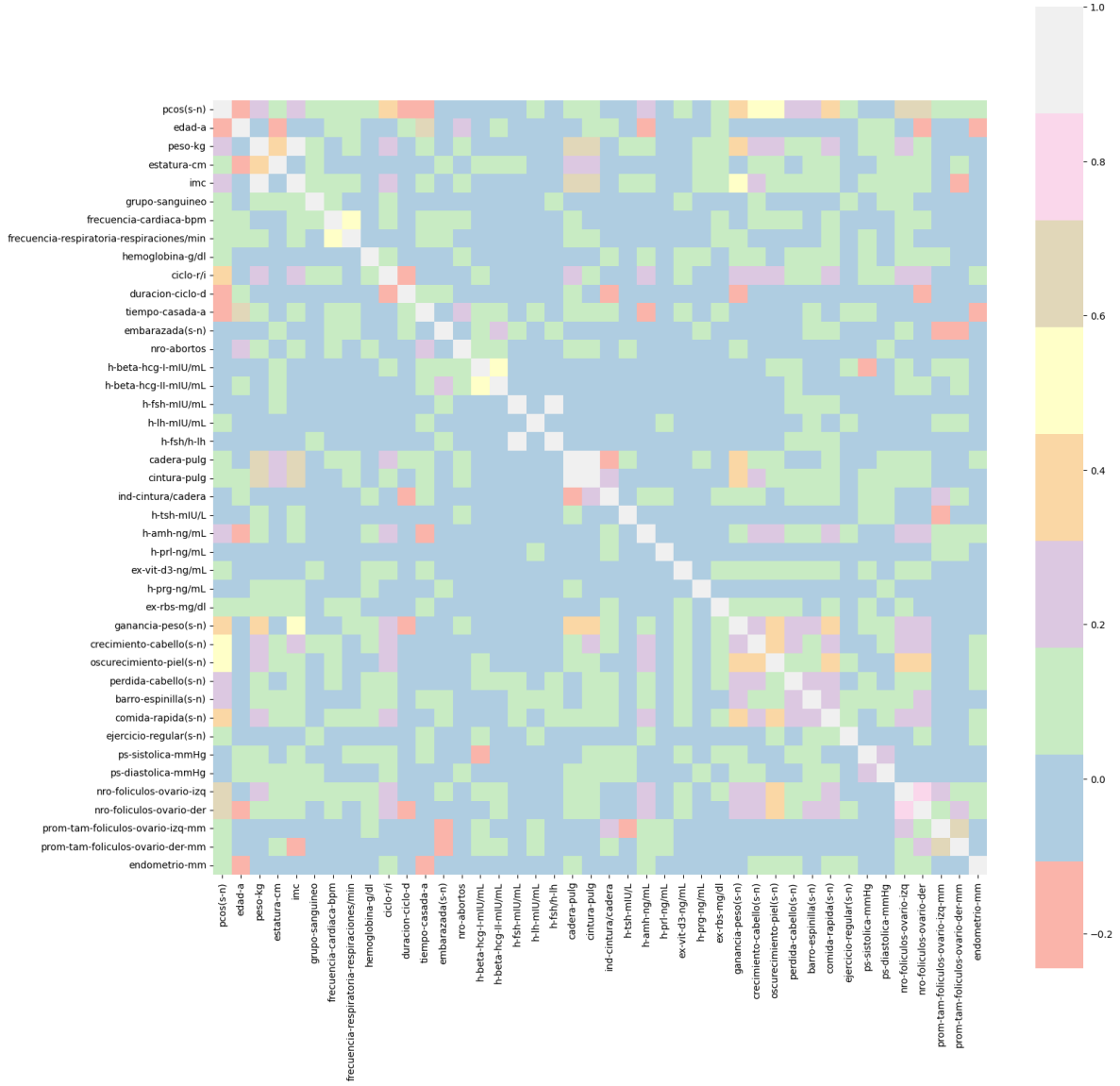
In [ ]:
```
# Grafico de caja sin outlier para ciclo regular o irregular
px.box(df['ciclo-r/i'], points='all')
```

Visualizacion de datos

In [ ]:
```
#Relacion entre las variables
sns.pairplot(df)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-94-19fb3040fc42> in <cell line: 2>()
      1 #Relacion entre las variables
----> 2 sns.pairplot(df)

/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py in pairplot(data, hue, hue_order, palette, vars, x_vars, y_vars, kind, diag_kind, markers, height, aspect, corner, dropna, plot_kws, diag_kws, grid_kws, size)
   2117     # Set up the PairGrid
   2118     grid_kws.setdefault("diag_sharey", diag_kind == "hist")
-> 2119     grid = PairGrid(data, vars=vars, x_vars=x_vars, y_vars=y_vars, hue=hue,
   2120                     hue_order=hue_order, palette=palette, corner=corner,
   2121                     height=height, aspect=aspect, dropna=dropna, **grid_kws)
```

In [ ]:
```
#Determinamos una matriz de correlación de todos los valores de cada parametro

# sacar las variables categoricas del grafico de correlacion.
corrmat = df.corr()
plt.subplots(figsize=(18,18))
sns.heatmap(corrmat,cmap="Pastel1", square=True);
```

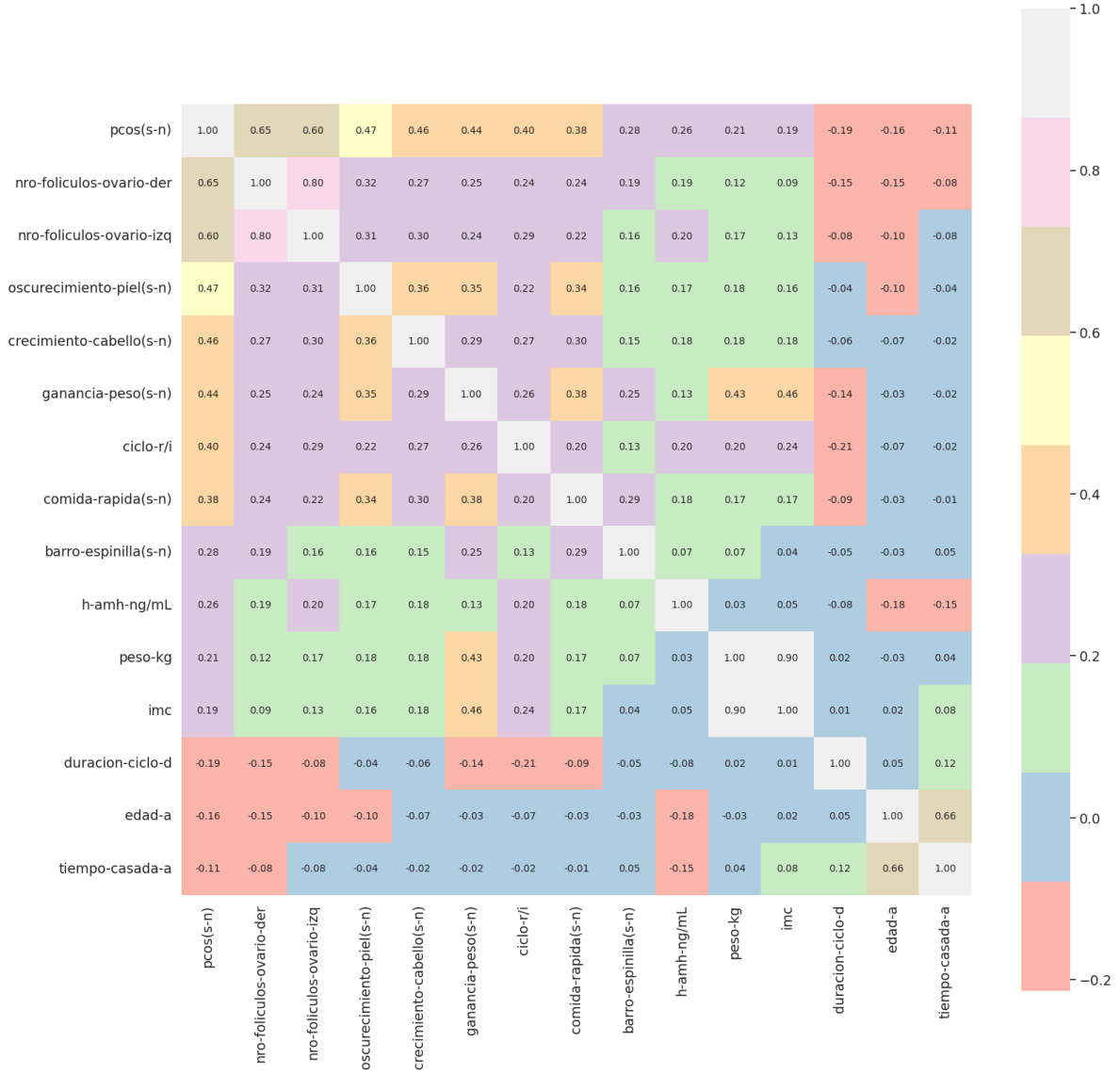


Analisis univariado y multivariado

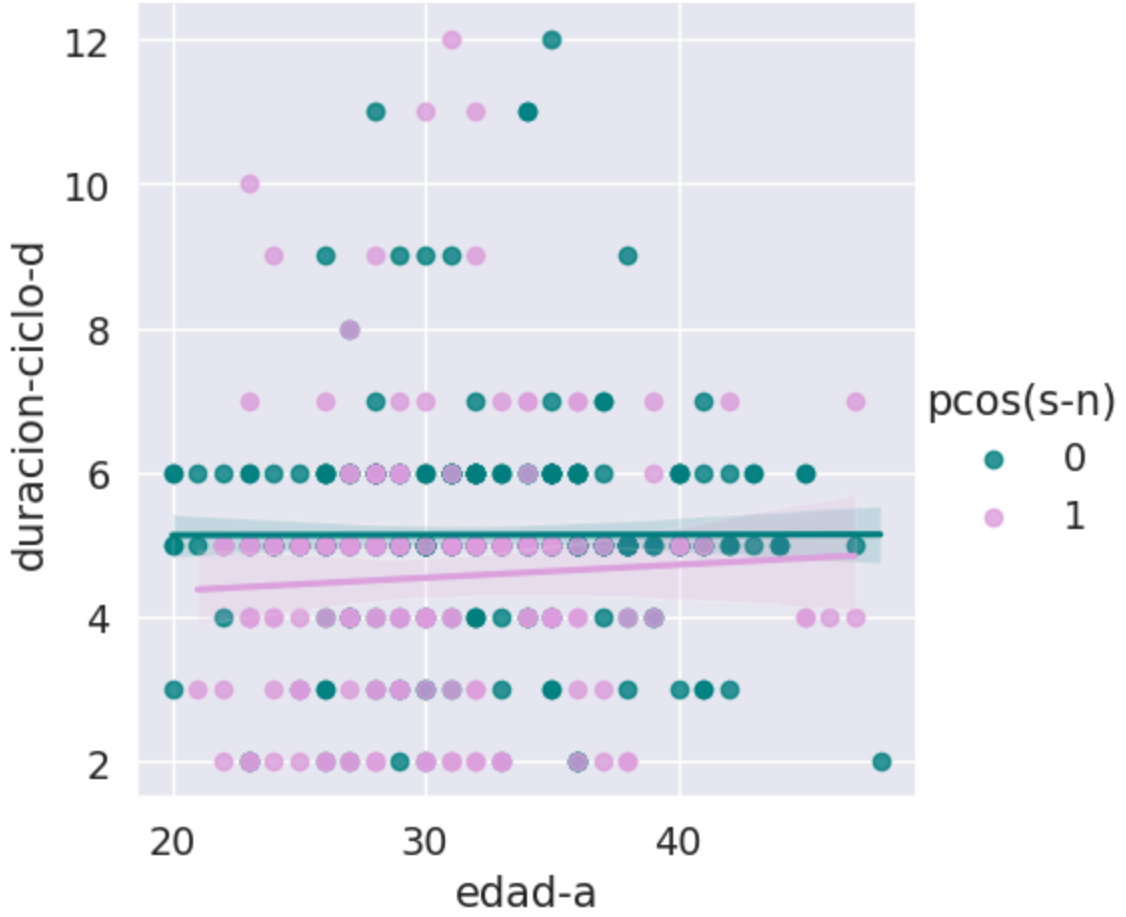
In [ ]:
```
# Determinamos el valor de correlacion de todos los parametros para el paramet
# desde el más significativo al menos
corrmat = df.corr()
corrmat['pcos(s-n)'].sort_values(ascending=False)
```

Out[96]:
```
pcos(s-n)                                  1.000000
nro-foliculos-ovario-der                   0.646351
nro-foliculos-ovario-izq                   0.599341
oscurecimiento-piel(s-n)                   0.474489
crecimiento-cabello(s-n)                   0.460212
ganancia-peso(s-n)                         0.439283
ciclo-r/i                                  0.398542
comida-rapida(s-n)                         0.376221
barro-espinilla(s-n)                       0.282038
h-amh-ng/mL                                0.261674
peso-kg                                    0.211253
imc                                        0.194579
perdida-cabello(s-n)                       0.169980
cadera-pulg                                0.162737
cintura-pulg                               0.159199
prom-tam-foliculos-ovario-izq-mm           0.103665
frecuencia-cardiaca-bpm                    0.099702
hemoglobina-g/dl                           0.093411
endometrio-mm                              0.093017
prom-tam-foliculos-ovario-der-mm           0.089039
ex-vit-d3-ng/mL                            0.085134
estatura-cm                                0.076798
ejercicio-regular(s-n)                     0.066775
h-lh-mIU/mL                                0.063986
ex-rbs-mg/dl                               0.052260
grupo-sanguineo                            0.036001
frecuencia-respiratoria-respiraciones/min  0.035306
ps-diastolica-mmHg                         0.024524
h-beta-hcg-II-mIU/mL                       0.010797
ind-cintura/cadera                        -0.000123
h-prl-ng/mL                               -0.004323
h-tsh-mIU/L                               -0.011349
h-fsh/h-lh                                -0.019467
ps-sistolica-mmHg                         -0.020417
h-beta-hcg-I-mIU/mL                       -0.030657
h-fsh-mIU/mL                              -0.031089
embarazada(s-n)                           -0.032477
h-prg-ng/mL                               -0.045387
nro-abortos                               -0.053793
tiempo-casada-a                           -0.110596
edad-a                                    -0.164222
duracion-ciclo-d                          -0.188485
Name: pcos(s-n), dtype: float64
```

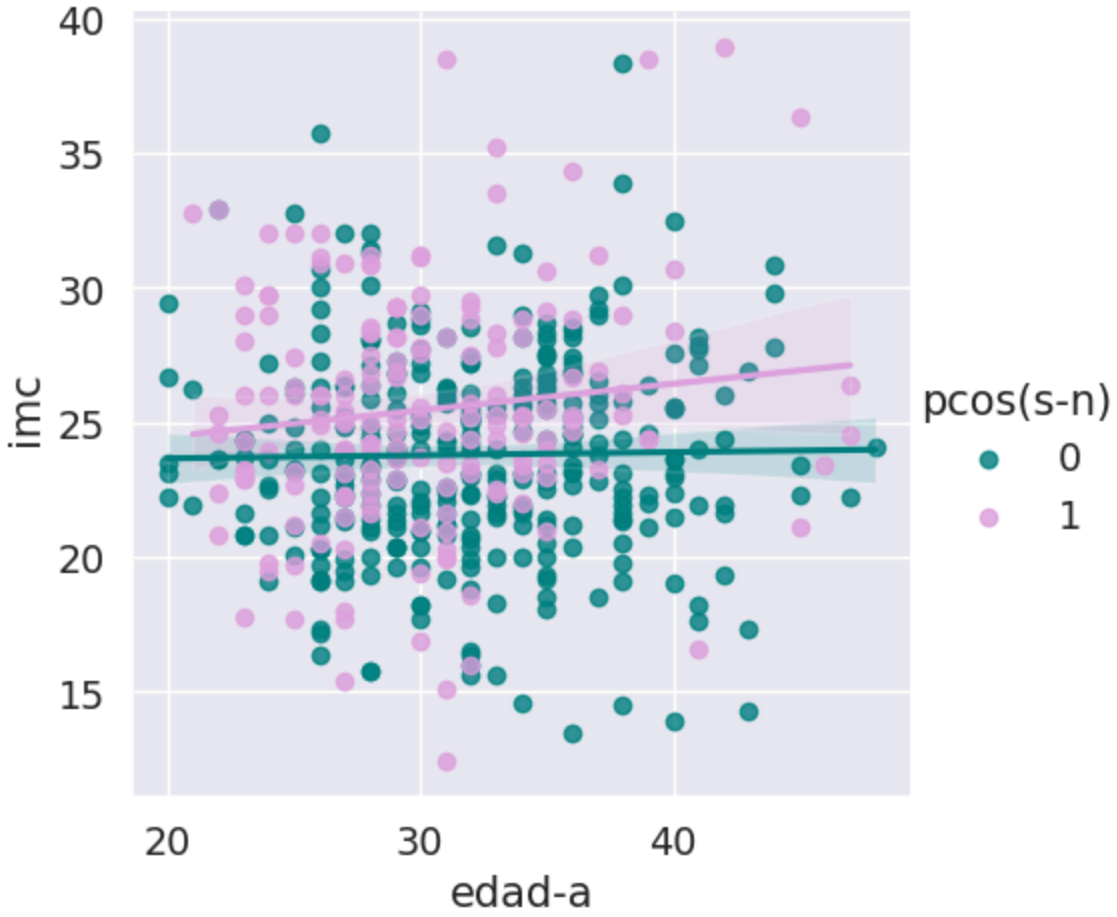
In [ ]:
```
#Bastantes variables tienen correlación significativa

plt.figure(figsize=(18,18))
k = 12 #número de variables con mapa de calor positivo
l = 3 #número de variables con mapa de calor negativo
cols_p = corrmat.nlargest(k, "pcos(s-n)")["pcos(s-n)"].index
cols_n = corrmat.nsmallest(l, "pcos(s-n)")["pcos(s-n)"].index
cols = cols_p.append(cols_n)

cm = np.corrcoef(df[cols].values.T)
sns.set(font_scale=1.25)
hm = sns.heatmap(cm, cbar=True,cmap="Pastel1", annot=True, square=True, fmt='.
                 annot_kws={'size': 10}, yticklabels=cols.values, xticklabels=
plt.show()
```



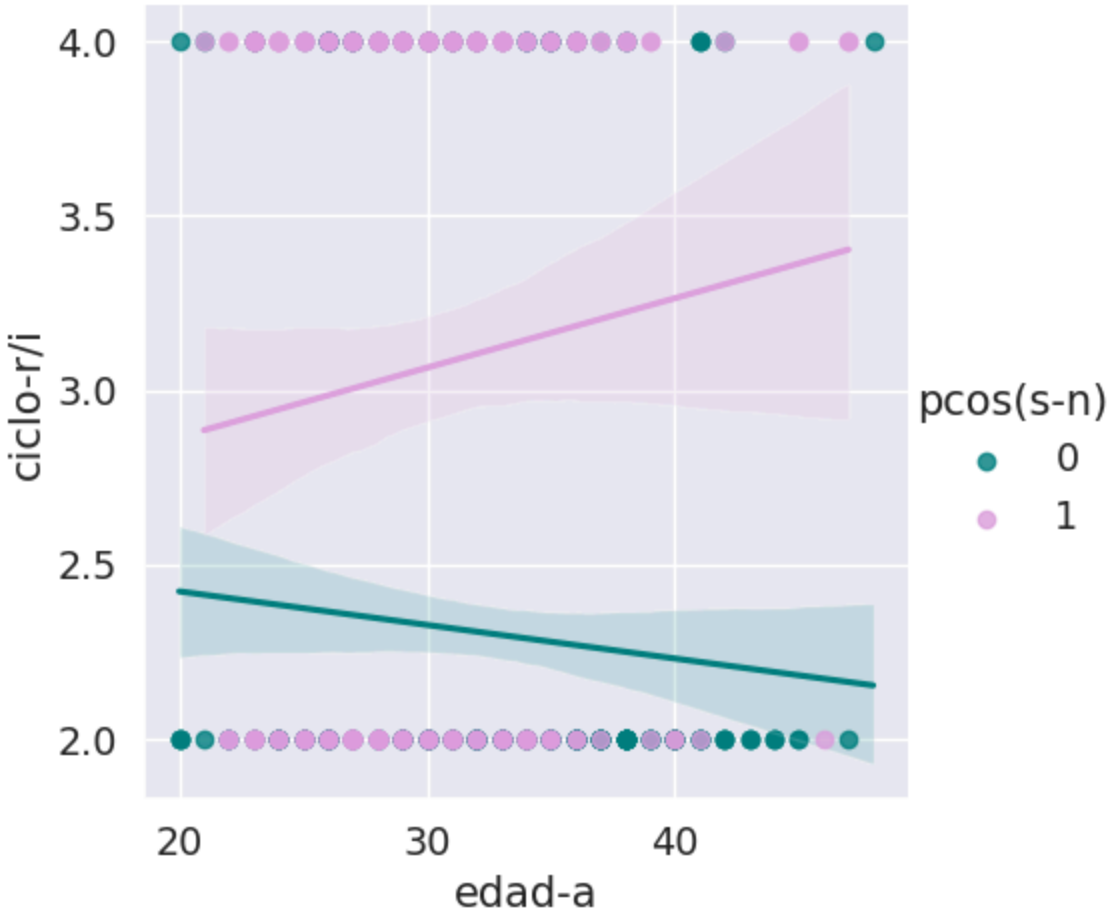
In [ ]:
```
#Duración de la fase menstrual en sindrome de ovario poliquistivo vs normal
color = ["teal", "plum"]
figura13 = sns.lmplot(data=df,x="edad-a",y="duracion-ciclo-d", hue="pcos(s-n)"
plt.show(figura13)
```



In [ ]:
```
# Patrón de aumento de peso (IMC) a lo largo de los años en pacientes con pcos
figura14= sns.lmplot(data =df,x="edad-a",y="imc", hue="pcos(s-n)", palette= co
plt.show(figura14)
```



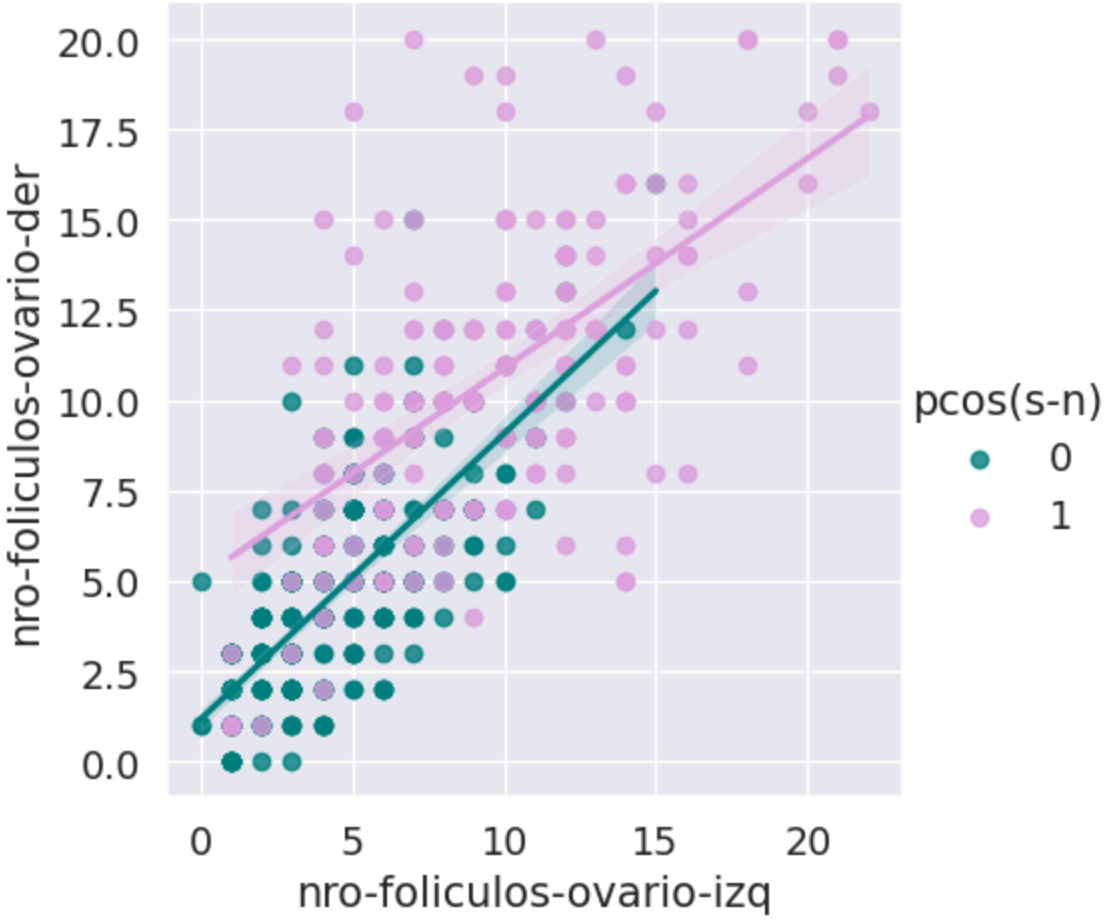
In [ ]:
```
# Patron del ciclo menstrual (r/i) con la edad y el pcos
sns.lmplot(data =df,x="edad-a",y="ciclo-r/i", hue="pcos(s-n)",palette=color)
plt.show()
```



Aunque no es una convencíon después del análisis de evidencia que probablemente el tipo de respuesta igual a 2 equivale a un ciclo irregular y 4 a un ciclo regular.

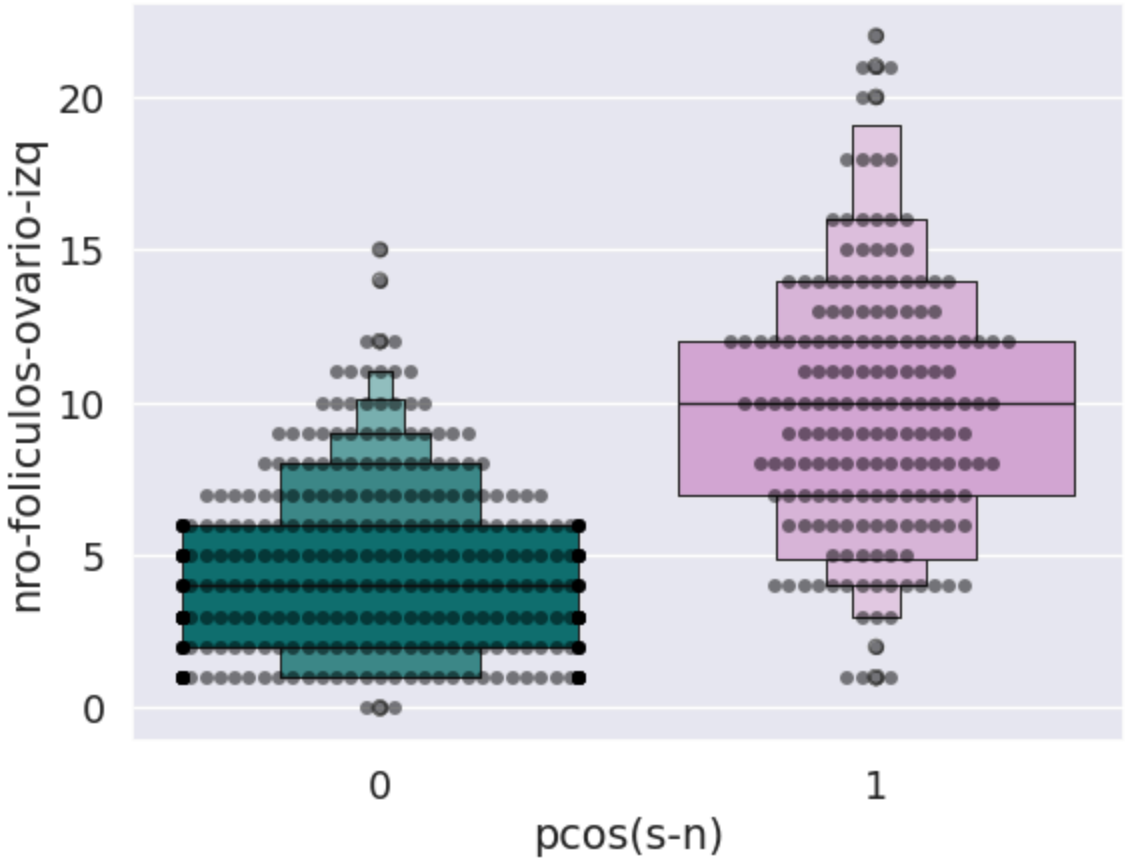
ciclo-r/i = 2 para ciclos irregulares ciclo-r/i = 4 para ciclos regulares

In [ ]:
```
# Distribución de folicutlos en ambos ovarios
sns.lmplot(data =df,x='nro-foliculos-ovario-izq',y='nro-foliculos-ovario-der',
           hue='pcos(s-n)',palette=color)
plt.show()
```

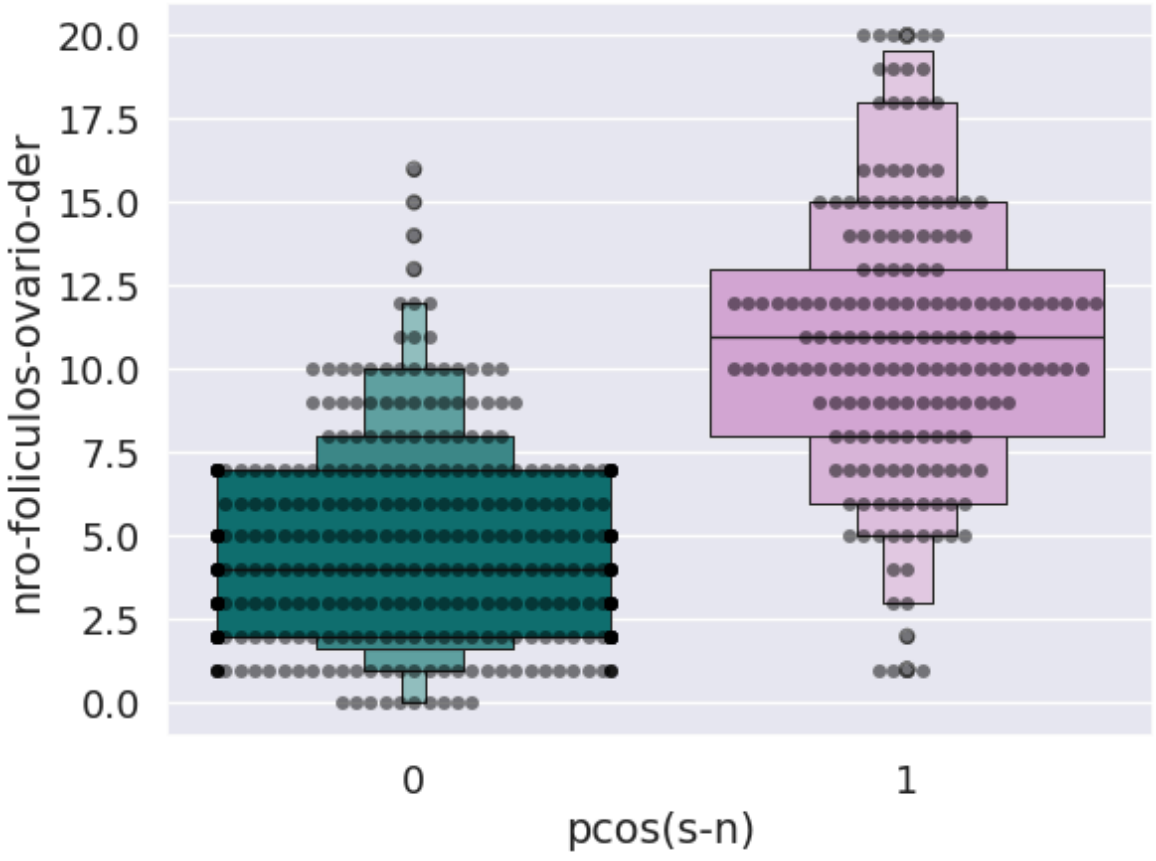


In [ ]:
```
caracteristicas_fol = ["nro-foliculos-ovario-izq","nro-foliculos-ovario-der"]
for i in caracteristicas_fol:
    sns.swarmplot(x=df["pcos(s-n)"], y=df[i], color="black", alpha=0.5 )
    sns.boxenplot(x=df["pcos(s-n)"], y=df[i], palette=color)
    plt.show()
```

```
<ipython-input-103-8f5a63c22329>:4: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be removed i
n v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the sa
me effect.


/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3398: UserWarn
ing:

32.0% of the points cannot be placed; you may want to decrease the size of th
e markers or use stripplot.
```

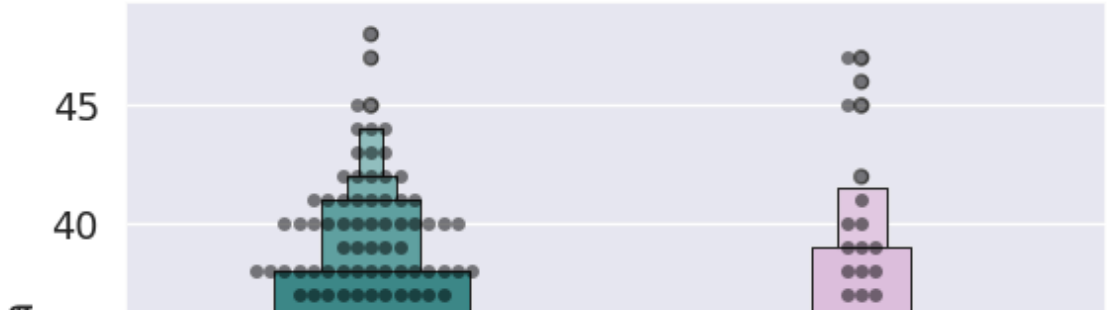


```
<ipython-input-103-8f5a63c22329>:4: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be removed i
n v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the sa
me effect.


/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3398: UserWarn
ing:

28.3% of the points cannot be placed; you may want to decrease the size of th
e markers or use stripplot.
```



In [ ]:
```
caracteristicas = ['edad-a', 'peso-kg', 'imc', 'hemoglobina-g/dl', 'duracion-c
for i in caracteristicas:
    sns.swarmplot(x=df['pcos(s-n)'], y=df[i], color="black", alpha=0.5 )
    sns.boxenplot(x=df['pcos(s-n)'], y=df[i], palette=color)
    plt.show()
```

```
<ipython-input-104-8c6d0ef402e0>:4: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be remove
d in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for
the same effect.
```



Conclusiones

La duración de la fase menstrual es, en general, constante en diferentes edades en los casos normales. Mientras que en el caso del PCOS la duración aumentó con la edad.

El índice de masa corporal (IMC) muestra consistencia en los casos normales. Mientras que en el caso del síndrome de ovario poliquístico, el IMC aumenta con la edad.

El ciclo menstrual se vuelve más regular en los casos normales con la edad. Mientras que en el caso del síndrome de ovario poliquístico la irregularidad aumenta con la edad.

La distribución de los folículos en ambos ovarios, izquierdo y derecho, no es igual para las mujeres con síndrome de ovario poliquístico en comparación con la paciente "normal".

La cantidad de folículos en mujeres con pcos es mayor, como se esperaba. Y también son desiguales.

La muestra es corta para determinar si hay hechos patognómonicas que arrojen nueva información a los estudios y conociemientos actuales. Se recomienda aumentar la muestra.

###4. Modelado

Evaluación de Modelos de Clasificación para Predicción del Síndrome de Ovarios Poliquísticos

Para abordar el objetivo de este ejercicio, que consiste en desarrollar un modelo capaz de predecir si una mujer es propensa a padecer síndrome de ovarios poliquísticos, se evaluarán inicialmente un total de ocho modelos de clasificación. Estos modelos son:

- Regresión Logística
- KNN
- Naive Bayes
- Árboles de Decisión con criterio ID3
- Árboles de Decisión con criterio CART
- Bagging
- Random Forest
- XGBoost

Escalado de los Datos

Dadas las diferentes escalas de las variables, se opta por realizar una estandarización de los datos. Este proceso escrucial para garantizar que todas las variables contribuyan de manera equitativa al modelo, evitando que aquellas con mayor magnitud dominen el proceso de aprendizaje. El escalado de datos mejora la eficiencia y el rendimiento del modelo, asegurando una mejor convergencia durante el entrenamiento y permitiendo una comparación más justa entre las diferentes variables.

Métricas Clave

Accuracy: Esta métrica permite determinar la precisión del modelo en términos de aciertos tanto en mujeres que tienen como en las que no tienen el síndrome de ovarios poliquísticos.

Recall: Dado que es crucial minimizar la cantidad de falsos negativos (mujeres a las que se les predice que no tienen el síndrome cuando en realidad sí lo tienen), el recall es esencial para evaluar la capacidad del modelo para identificar correctamente los casos positivos.

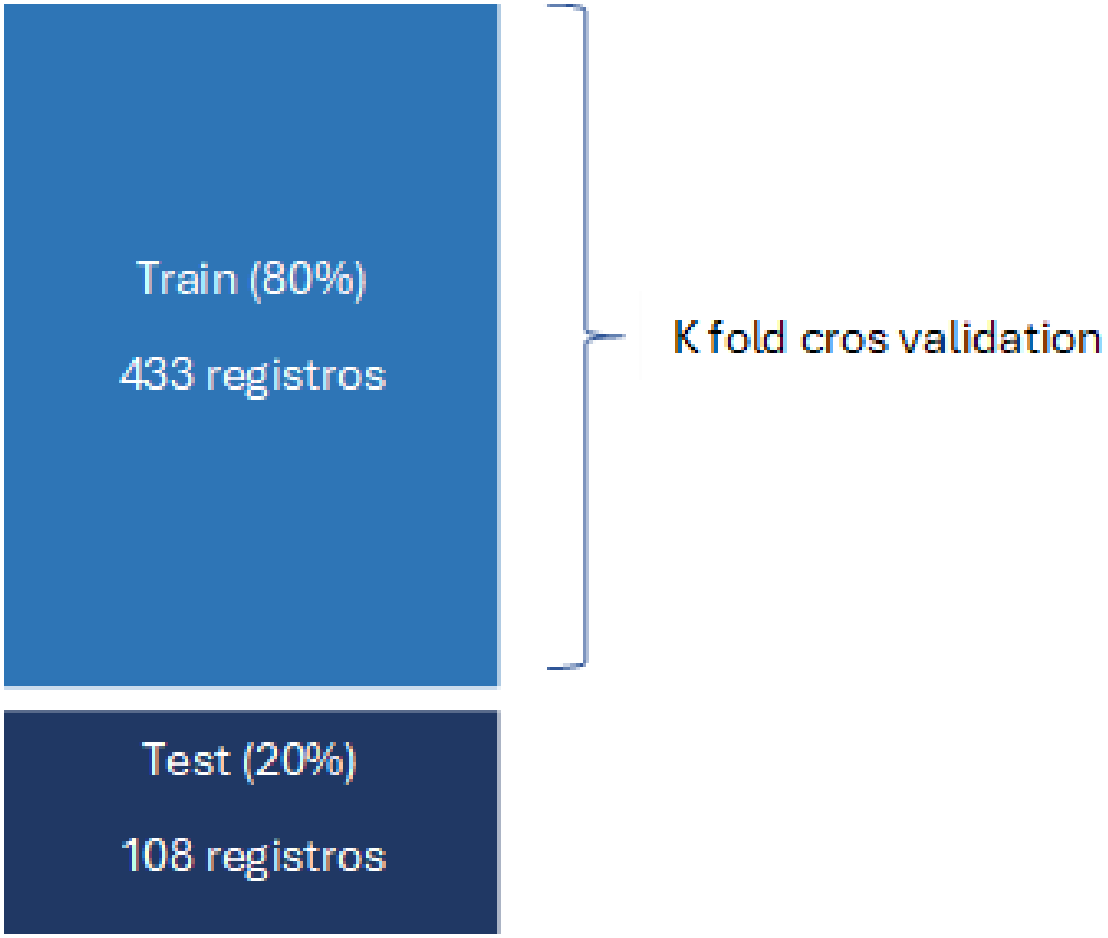
Protocolos de Evaluación

Para obtener una evaluación más completa y llegar al modelo óptimo, cada uno de los modelos se someterá a dos fases de análisis:

1. **Fase sin búsqueda de hiperparámetros:** En esta fase inicial, los modelos se evaluarán con sus configuraciones por defecto.
2. **Fase con búsqueda de hiperparámetros:** En la segunda fase, se realizará una optimización de hiperparámetros utilizando los protocolos de validación cruzada K-fold y GridSearchCV.

Esquema del Proceso

El esquema gráfico del proceso es el siguiente:



Selección del Modelo

1. **Selección del modelo:** Evaluar todos los modelos listados y calcular las métricas respectivas (accuracy y recall).
2. **Optimización del modelo seleccionado:** Seleccionar el modelo con mejor rendimiento y realizar una búsqueda de hiperparámetros más exhaustiva utilizando GridSearchCV y K-fold cross validation para obtener el modelo definitivo.

Este enfoque nos permitirá identificar el modelo con mejor desempeño en la predicción del

In [ ]:
```
# Codificando en 1 y 0 ciclo regular
df["ciclo-r/i"]=df["ciclo-r/i"].replace({4:1,2:0})
df.rename(columns={"ciclo-r/i":"ciclo-r"}, inplace=True)
```

Modelo 1: Regresión Logística

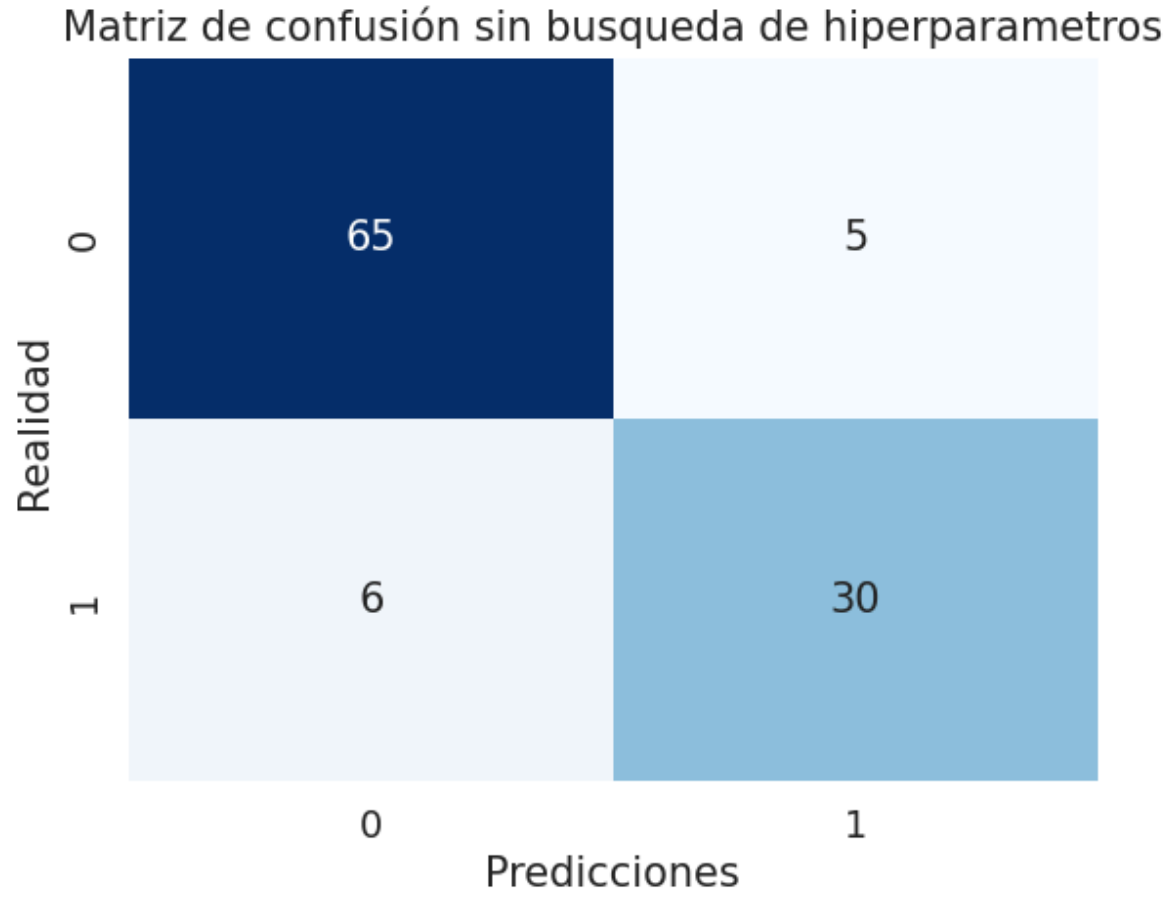
Presentamos versiones de cada modelo, uno sin busqueda de hiérparametros y otro con busqueda.

Regresión Logística sin busqueda de hiperperparametros óptimos

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, classification_r
from sklearn.model_selection import GridSearchCV, KFold
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
import seaborn as sns


###Modelo de regresión Logisitica


# Preparar los datos
# Separar las características (X) de la variable objetivo (y)
X = df.drop(columns=['pcos(s-n)'])
y = df['pcos(s-n)']

# Escalar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Paso 3: Dividir el dataset en conjunto de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2

# Entrenar el modelo de regresión logística con un mayor número de iteraciones
model = LogisticRegression(max_iter=5000, solver='lbfgs')
model.fit(X_train, y_train)

# Evaluar el modelo
y_pred_rl = model.predict(X_test)

# Calcular la precisión
accuracy = accuracy_score(y_test, y_pred_rl)
print(f'Accuracy: {accuracy:.2f}')

# Matriz de confusión
conf_matrix_rl = confusion_matrix(y_test, y_pred_rl)
print('Matriz de confusión:')
print(conf_matrix_rl)

# Reporte de clasificación
class_report_rl = classification_report(y_test, y_pred_rl)
print('Reporte:')
print(class_report_rl)

recall_class_rl = recall_score(y_test, y_pred_rl, pos_label=0)

# Crear Matriz
sns.heatmap(conf_matrix_rl, annot=True, fmt='d', cmap='Blues', cbar=False)

# Configurar las etiquetas de los ejes y el título
plt.xlabel('Predicciones')
plt.ylabel('Realidad')
plt.title('Matriz de confusión sin busqueda de hiperparametros')
```

```
# Mostrar la visualización
plt.show()
```

```
Accuracy: 0.90
Matriz de confusión:
[[65  5]
 [ 6 30]]
Reporte:
              precision    recall  f1-score   support

           0       0.92      0.93      0.92        70
           1       0.86      0.83      0.85        36

    accuracy                           0.90       106
   macro avg       0.89      0.88      0.88       106
weighted avg       0.90      0.90      0.90       106
```

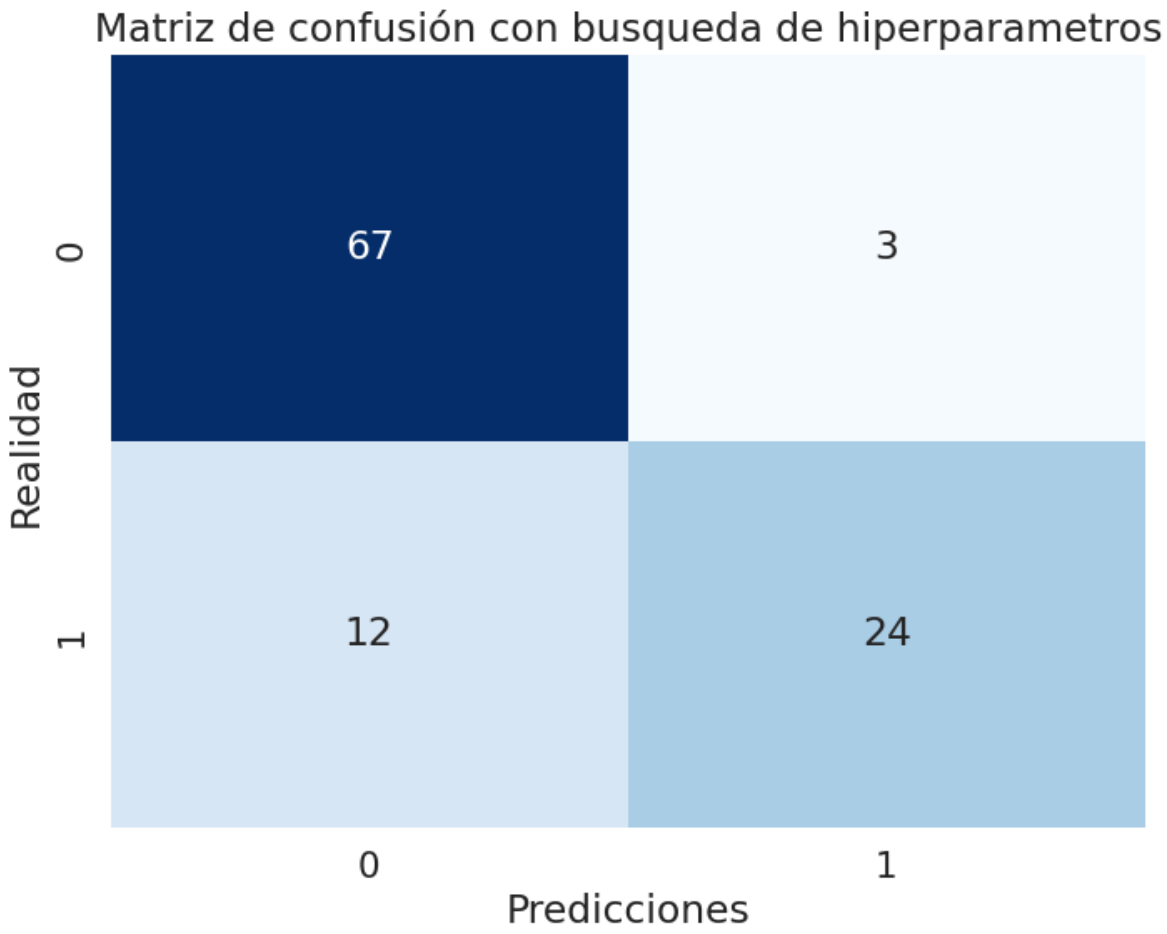


Regresión Logística con busqueda de hiperperparametros óptimos

In [ ]:
```
# Definir los hiperparámetros a ajustar
param_grid = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'penalty': ['l2', 'none']
}

# Definir el protocolo de evaluación K-fold
kfold = KFold(n_splits=5, shuffle=True, random_state=42)

# Inicializar el GridSearchCV
grid_search = GridSearchCV(estimator=LogisticRegression(max_iter=5000, solver=
                           param_grid=param_grid,
                           cv=kfold,
                           scoring='accuracy',
                           verbose=1,
                           n_jobs=-1)

# Entrenar el modelo utilizando GridSearchCV
grid_search.fit(X_train, y_train)

# Mejores hiperparámetros encontrados
best_params = grid_search.best_params_
print("Best Parameters:", best_params)

# Mejor modelo
best_model = grid_search.best_estimator_


cv_scores = cross_val_score(best_model, X_test, y_test, cv=kfold)
print("Cross Validation Scores:", cv_scores)

# Calcular la precisión media de la validación cruzada
mean_accuracy = cv_scores.mean()
print(f'Mean Accuracy (Cross Validation): {mean_accuracy:.2f}')


# Evaluar utilizando recall
cv_scores_recall = cross_val_score(best_model, X_test, y_test, cv=kfold, scori
print("Cross Validation Recall Scores:", cv_scores_recall)

# Calcular el recall medio de la validación cruzada
mean_recall = cv_scores_recall.mean()
print(f'Mean Recall (Cross Validation): {mean_recall:.2f}')

#Matriz de confusión
y_pred_cv = cross_val_predict(best_model, X_test, y_test, cv=kfold)

# Calcular la matriz de confusión
conf_matrix = confusion_matrix(y_test, y_pred_cv)
print('Confusion Matrix:')
print(conf_matrix)


# Crear una figura y un eje (axis) para la visualización
plt.figure(figsize=(8, 6))
sns.set(font_scale=1.4)  # Ajustar el tamaño de fuente
```

```
class_report_rl_2 = classification_report(y_test, y_pred_cv)
print('Reporte:')
print(class_report_rl_2)

# Crear el heatmap
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)

# Configurar las etiquetas de los ejes y el título
plt.xlabel('Predicciones')
plt.ylabel('Realidad')
plt.title('Matriz de confusión con busqueda de hiperparametros')

# Mostrar la visualización
plt.show()
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
Best Parameters: {'C': 0.1, 'penalty': 'l2'}
Cross Validation Scores: [0.90909091 0.85714286 0.85714286 0.9047619  0.76190
476]
Mean Accuracy (Cross Validation): 0.86
Cross Validation Recall Scores: [0.8        0.83333333 0.625      0.71428571
0.2       ]
Mean Recall (Cross Validation): 0.63
Confusion Matrix:
[[67  3]
 [12 24]]
Reporte:
              precision    recall  f1-score   support

           0       0.85      0.96      0.90        70
           1       0.89      0.67      0.76        36

    accuracy                           0.86       106
   macro avg       0.87      0.81      0.83       106
weighted avg       0.86      0.86      0.85       106
```



Conclusión:

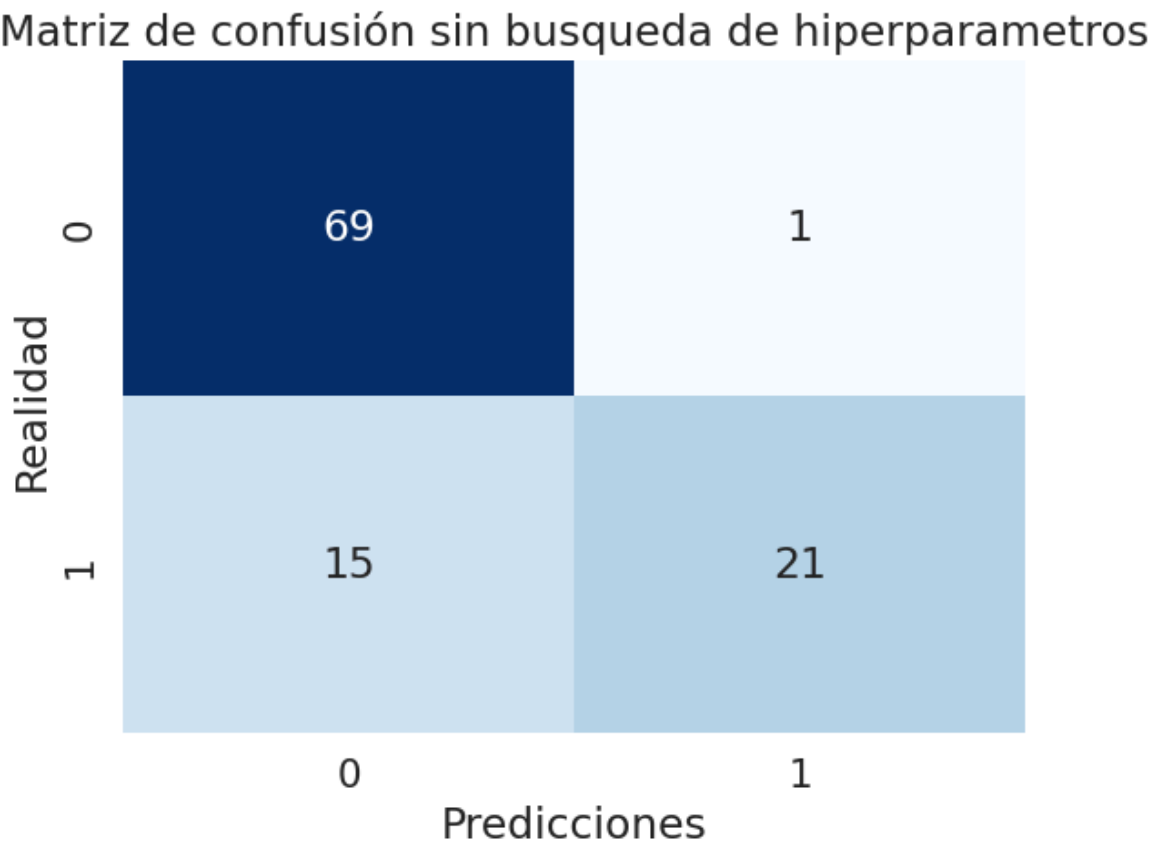
Al corres el modelo de regresión logística y el modelo con búsqueda de hiperparámetros se puede evidenciar que el segundo modelo es arroja un mejor resultado más preciso pasando de un 85% a un 88%, indicando en cuanto a las predicciones es mucho mas acertado.

Modelo: K vecinos más cercanos

K vecinos más cercanos sin busqueda de hiperparametros

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier

#Modelo KNN

k = 5
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)

# Evaluar el modelo
y_pred_knn = knn.predict(X_test)

# Calcular la precisión
accuracy_knn = accuracy_score(y_test, y_pred_knn)
print(f'Accuracy: {accuracy_knn:.2f}')

# Matriz de confusión
conf_matrix_knn = confusion_matrix(y_test, y_pred_knn)
print('Matriz de confusión:')
print(conf_matrix_knn)

# Reporte de clasificación
class_report_knn = classification_report(y_test, y_pred_knn)
print('Reporte:')
print(class_report_knn)

# Crear Matriz
sns.heatmap(conf_matrix_knn, annot=True, fmt='d', cmap='Blues', cbar=False)

# Configurar las etiquetas de los ejes y el título
plt.xlabel('Predicciones')
plt.ylabel('Realidad')
plt.title('Matriz de confusión sin busqueda de hiperparametros')

# Mostrar la visualización
plt.show()
```

```
Accuracy: 0.85
Matriz de confusión:
[[69  1]
 [15 21]]
Reporte:
              precision    recall  f1-score   support

           0       0.82      0.99      0.90        70
           1       0.95      0.58      0.72        36

    accuracy                           0.85       106
   macro avg       0.89      0.78      0.81       106
weighted avg       0.87      0.85      0.84       106
```

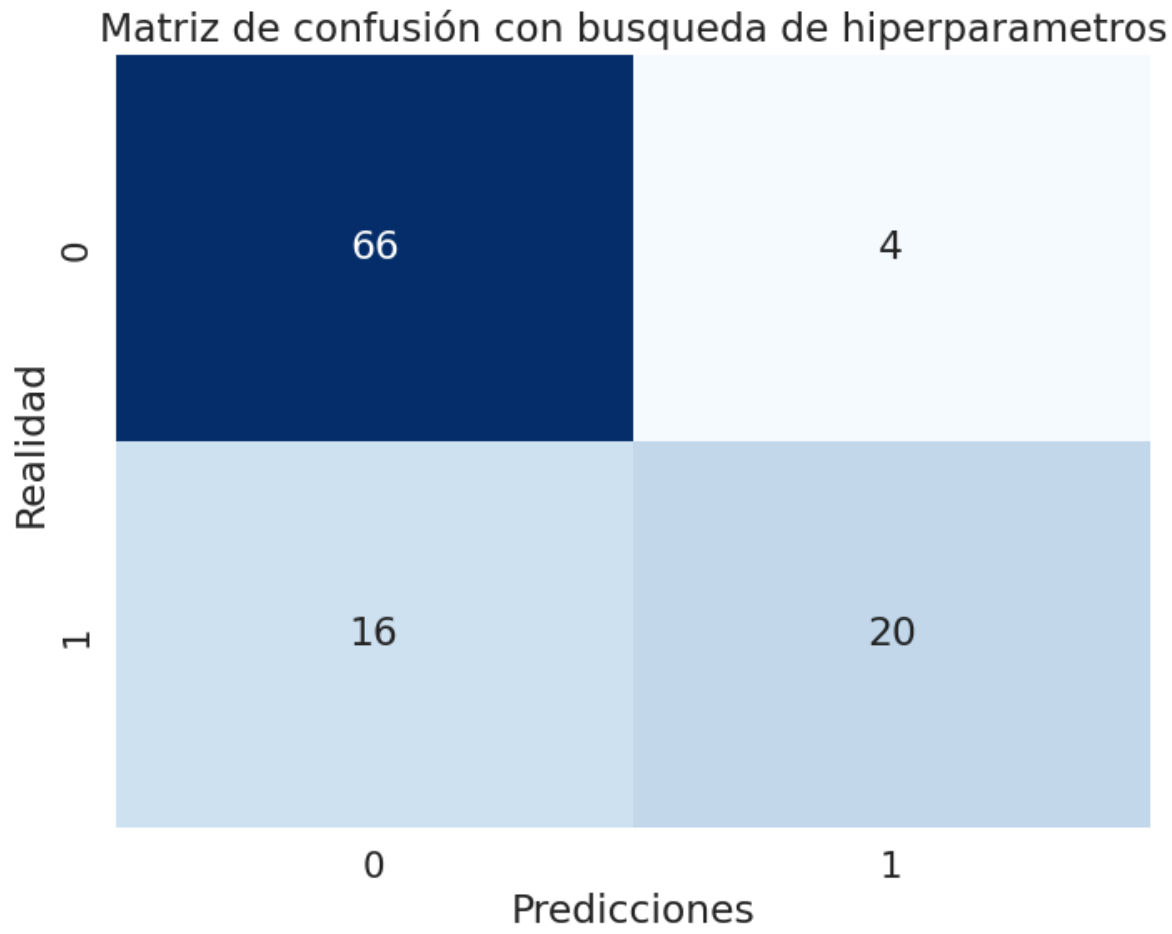


K vecinos más cercanos con busqueda de hiperparametros

In [ ]:
```
# Definir los hiperparámetros a ajustar
param_grid = {
    'n_neighbors': [3, 5, 7, 9, 11],  # Número de vecinos
    'weights': ['uniform', 'distance'],  # Test pesos uniformes
    'metric': ['euclidean', 'manhattan']  # Test distancias
}

# Definir el protocolo de evaluación K-fold
kfold = KFold(n_splits=5, shuffle=True, random_state=42)

# Inicializar el GridSearchCV
grid_search = GridSearchCV(estimator=KNeighborsClassifier(),
                           param_grid=param_grid,
                           cv=kfold,
                           scoring='accuracy',
                           verbose=1,
                           n_jobs=-1)

# Entrenar el modelo utilizando GridSearchCV
grid_search.fit(X_train, y_train)

# Mejores hiperparámetros encontrados
best_params_knn = grid_search.best_params_
print("Best Parameters:", best_params_knn)

# Mejor modelo
best_model = grid_search.best_estimator_


cv_scores_knn = cross_val_score(best_model, X_train, y_train, cv=kfold)
print("Cross Validation Scores:", cv_scores_knn)

# Calcular la precisión media de la validación cruzada
mean_accuracy = cv_scores.mean()
print(f'Mean Accuracy (Cross Validation): {mean_accuracy:.2f}')

#Matriz de confusión
y_pred_cv_knn = cross_val_predict(best_model, X_test, y_test, cv=kfold)

# Calcular la matriz de confusión
conf_matrix_knn = confusion_matrix(y_test, y_pred_cv_knn)
print('Confusion Matrix:')
print(conf_matrix_knn)


# Crear una figura y un eje (axis) para la visualización
plt.figure(figsize=(8, 6))
sns.set(font_scale=1.4)  # Ajustar el tamaño de fuente


class_report_knn_2 = classification_report(y_test, y_pred_cv_knn)
print('Reporte:')
print(class_report_knn_2)

# Crear el heatmap
sns.heatmap(conf_matrix_knn, annot=True, fmt='d', cmap='Blues', cbar=False)
```

```
# Configurar las etiquetas de los ejes y el título
plt.xlabel('Predicciones')
plt.ylabel('Realidad')
plt.title('Matriz de confusión con busqueda de hiperparametros')

# Mostrar la visualización
plt.show()
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
Best Parameters: {'metric': 'manhattan', 'n_neighbors': 5, 'weights': 'unifor
m'}
Cross Validation Scores: [0.83333333 0.91666667 0.91666667 0.89285714 0.80952
381]
Mean Accuracy (Cross Validation): 0.86
Confusion Matrix:
[[66  4]
 [16 20]]
Reporte:
              precision    recall  f1-score   support

           0       0.80      0.94      0.87        70
           1       0.83      0.56      0.67        36

    accuracy                           0.81       106
   macro avg       0.82      0.75      0.77       106
weighted avg       0.81      0.81      0.80       106
```

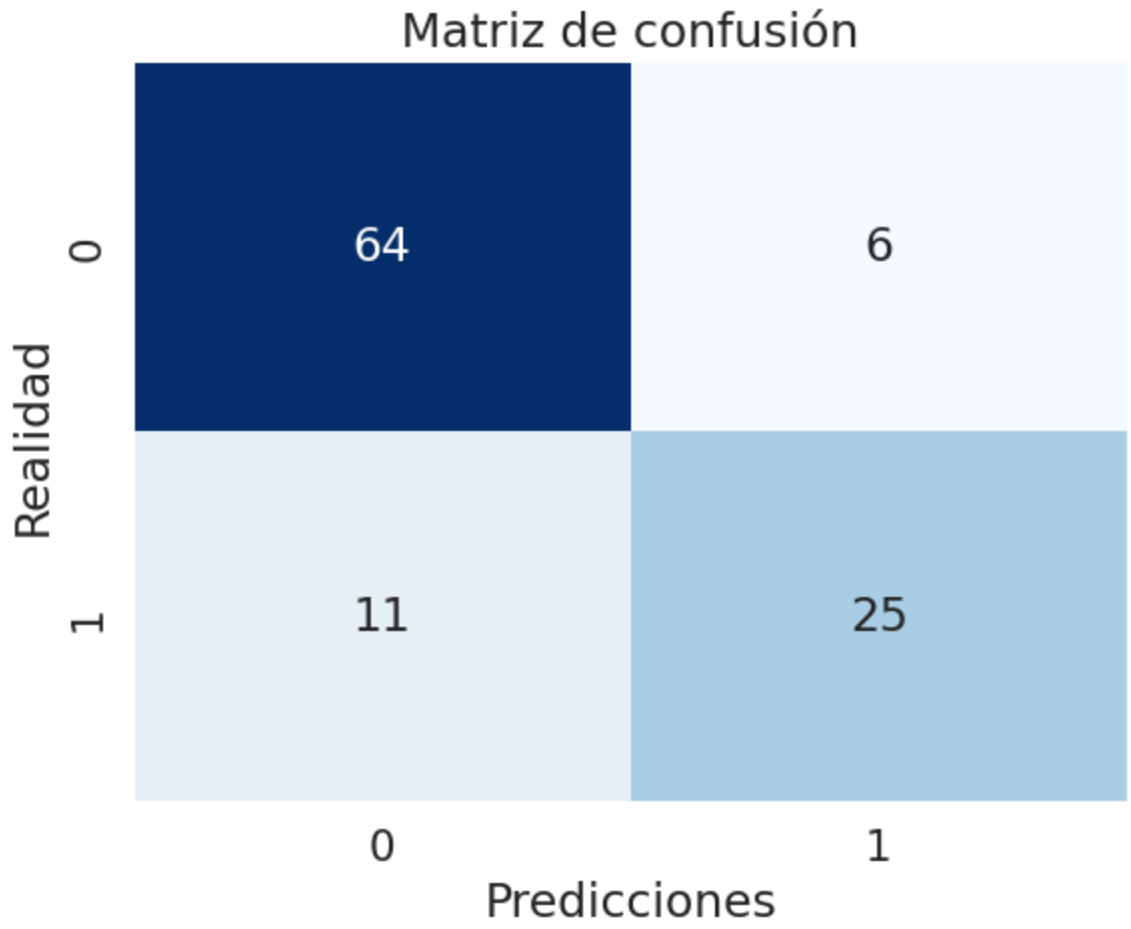


###Conclusión: Al corres el modelo de KNN y el modelo con búsqueda de hiperparámetros se puede evidenciar que el segundo modelo es arroja un mejor resultado más preciso pasando de un 84% a un 88%, indicando en cuanto a las predicciones es mucho mas acertado.

Modelo 3: Naive Bayes

In [ ]:
```
from sklearn.naive_bayes import GaussianNB
# Modelo Naive Bayes

# Entrenar el modelo Naive Bayes

nb = GaussianNB()
nb.fit(X_train, y_train)

# Evaluar el modelo
y_pred_nb = nb.predict(X_test)

# Calcular la precisión
accuracy_nb = accuracy_score(y_test, y_pred_nb)
print(f'Accuracy: {accuracy_nb:.2f}')

# Matriz de confusión
conf_matrix_nb = confusion_matrix(y_test, y_pred_nb)
print('Confusion Matrix:')
print(conf_matrix)

# Reporte de clasificación
class_report_NB = classification_report(y_test, y_pred_nb)
print('Classification Report:')
print(class_report_NB)

# Crear Matriz
sns.heatmap(conf_matrix_nb, annot=True, fmt='d', cmap='Blues', cbar=False)

# Configurar las etiquetas de los ejes y el título
plt.xlabel('Predicciones')
plt.ylabel('Realidad')
plt.title('Matriz de confusión')

# Mostrar la visualización
plt.show()
```

```
Accuracy: 0.84
Confusion Matrix:
[[67  3]
 [12 24]]
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.91      0.88        70
           1       0.81      0.69      0.75        36

    accuracy                           0.84       106
   macro avg       0.83      0.80      0.81       106
weighted avg       0.84      0.84      0.84       106
```



###Conclusión:

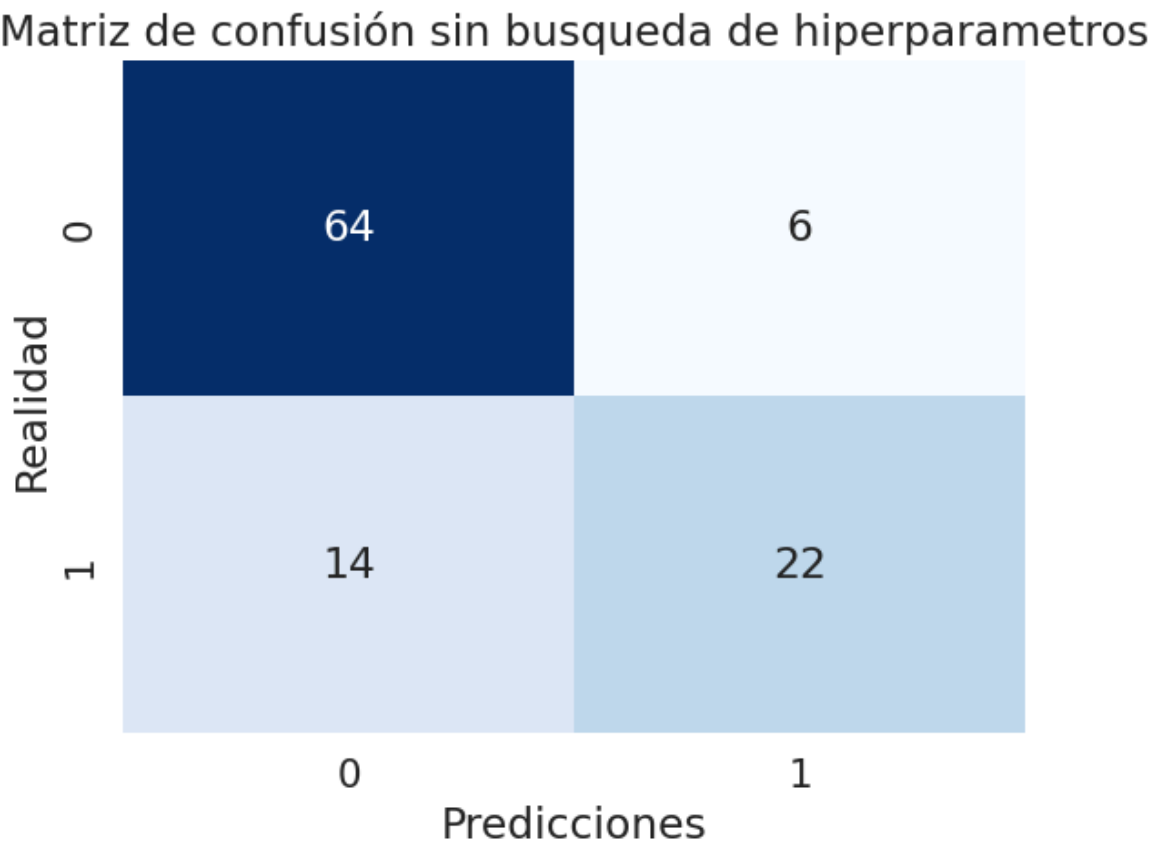
Al ejecutar el modelo nos arroja una exactitud del 78% y un recall 75% Indicando que el modelo puede tener cierta dificultad en identificar todas las instancias positivas correctamente.

Modelo 4: Arboles de decisión con criterio ID3

Arboles de decisión con criterio ID3 sin busqueda de hiperparametros

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier

### Arboles de desicion con criteriod ID3 (Entropia)

# Entrenar el modelo de árbol de decisión usando el criterio "entropy" (ID3)
dt = DecisionTreeClassifier(criterion='entropy', random_state=42)
dt.fit(X_train, y_train)

# Evaluar el modelo
y_pred_id3 = dt.predict(X_test)

# Calcular la precisión
accuracy_id3 = accuracy_score(y_test, y_pred_id3)
print(f'Accuracy: {accuracy_id3:.2f}')

# Matriz de confusión
conf_matrix_id3 = confusion_matrix(y_test, y_pred_id3)
print('Matriz de confusión:')
print(conf_matrix_id3)

# Reporte de clasificación
class_report_ID3 = classification_report(y_test, y_pred_id3)
print('Classification Report:')
print(class_report_ID3)

# Crear Matriz
sns.heatmap(conf_matrix_id3, annot=True, fmt='d', cmap='Blues', cbar=False)

# Configurar las etiquetas de los ejes y el título
plt.xlabel('Predicciones')
plt.ylabel('Realidad')
plt.title('Matriz de confusión sin busqueda de hiperparametros')

# Mostrar la visualización
plt.show()
```

```
Accuracy: 0.81
Matriz de confusión:
[[64  6]
 [14 22]]
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.91      0.86        70
           1       0.79      0.61      0.69        36

    accuracy                           0.81       106
   macro avg       0.80      0.76      0.78       106
weighted avg       0.81      0.81      0.80       106
```

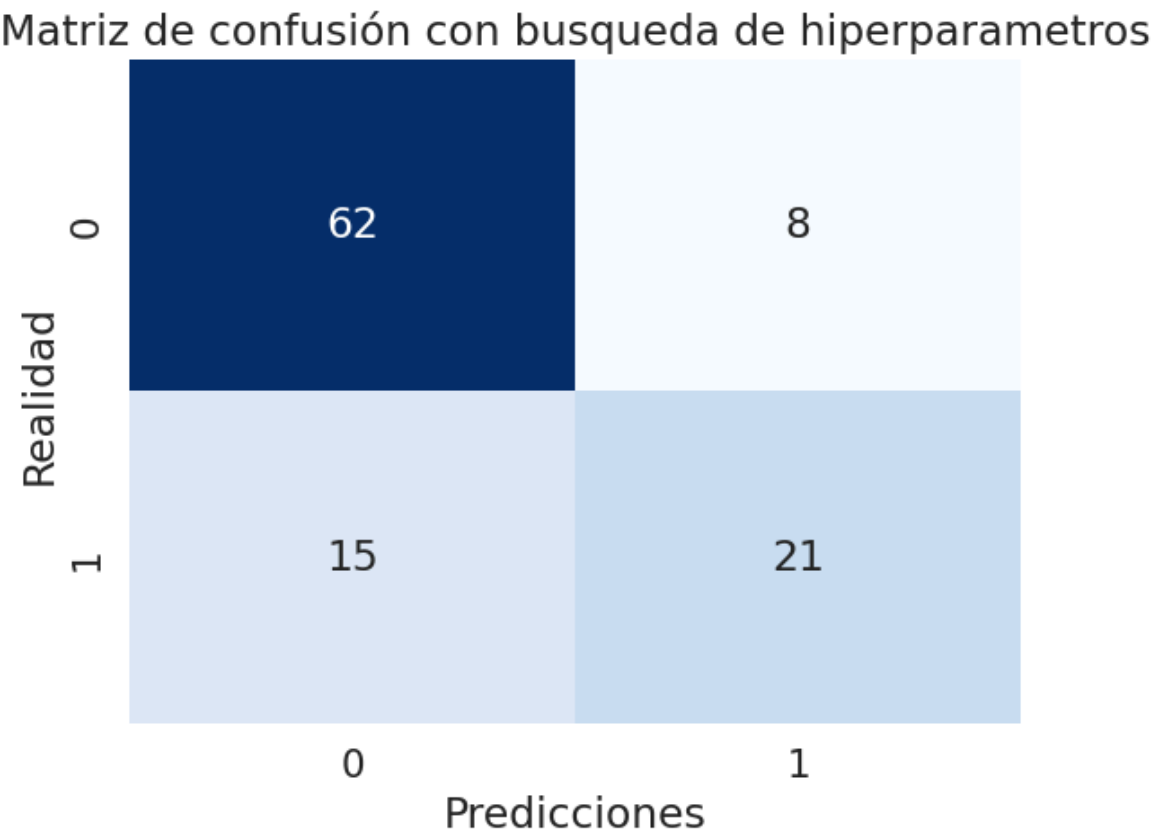


Arboles de decisión con criterio ID3 con busqueda de hiperparametros

In [ ]:
```
# Definir los hiperparámetros a ajustar
param_grid = {
    'max_depth': [None, 5, 10, 15],  # Vary the maximum depth of the tree
    'min_samples_split': [2, 5, 10],  # Vary the minimum number of samples req
    'min_samples_leaf': [1, 2, 4]  # Vary the minimum number of samples requir
}

# Definir el protocolo de evaluación K-fold
kfold = KFold(n_splits=5, shuffle=True, random_state=42)

# Inicializar el GridSearchCV
grid_search_id3 = GridSearchCV(estimator=DecisionTreeClassifier(criterion='ent
                               param_grid=param_grid,
                               cv=kfold,
                               scoring='accuracy',
                               verbose=1,
                               n_jobs=-1)

# Entrenar el modelo utilizando GridSearchCV
grid_search_id3.fit(X_train, y_train)

# Mejores hiperparámetros encontrados
best_params_id3 = grid_search.best_params_
print("Best Parameters:", best_params_id3)

# Mejor modelo
best_model_id3 = grid_search_id3.best_estimator_

y_pred_id3_2 = best_model_id3.predict(X_test)

# Calcular la precisión
accuracy_id3_2 = accuracy_score(y_test, y_pred_id3_2)
print(f'Accuracy: {accuracy_id3_2:.2f}')

# Matriz de confusión
conf_matrix_id3_2 = confusion_matrix(y_test, y_pred_id3_2)
print('Matriz de confusión:')
print(conf_matrix_id3_2)

# Reporte de clasificación
class_report_ID3_2 = classification_report(y_test, y_pred_id3_2)
print('Classification Report:')
print(class_report_ID3_2)

# Crear Matriz
sns.heatmap(conf_matrix_id3_2, annot=True, fmt='d', cmap='Blues', cbar=False)

# Configurar las etiquetas de los ejes y el título
plt.xlabel('Predicciones')
plt.ylabel('Realidad')
plt.title('Matriz de confusión con busqueda de hiperparametros')

# Mostrar la visualización
```

```
plt.show()
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
Best Parameters: {'metric': 'manhattan', 'n_neighbors': 5, 'weights': 'unifor
m'}
Accuracy: 0.78
Matriz de confusión:
[[62  8]
 [15 21]]
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.89      0.84        70
           1       0.72      0.58      0.65        36

    accuracy                           0.78       106
   macro avg       0.76      0.73      0.74       106
weighted avg       0.78      0.78      0.78       106
```



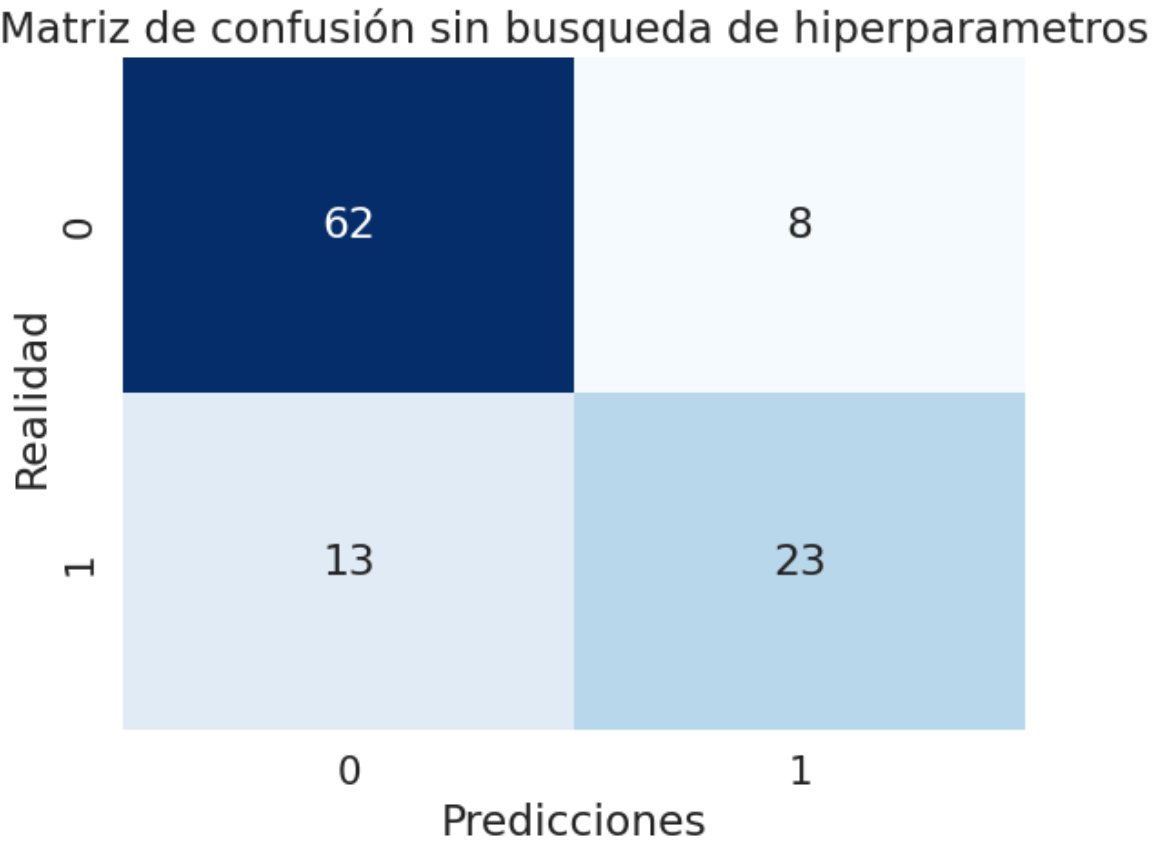
###Conclusión: En cuanto a los modelos de arboles de decisión con ID3 de entropía y el modelo con búsqueda de hiperparámetros los dos modelos arrojaron el mismo resultado de accuracy 79%, y tiene un recall de 83% y 87% respectivamente, nos indica que ambos modelos lograron un accuracy comparable, el modelo con búsqueda de hiperparámetros demostró una mejor capacidad para detectar los casos positivos

Modelo 5: Arboles de decisión con criterio CART

Arboles de decisión con criterio CART sin busqueda de hiperparametros

In [ ]:
```
# Entrenar el modelo de árbol de decisión usando el criterio "gini" (CART)
dt_cart = DecisionTreeClassifier(criterion='gini', random_state=42)
dt_cart.fit(X_train, y_train)

# Evaluar el modelo
y_pred_cart = dt_cart.predict(X_test)

# Calcular la precisión
accuracy_cart = accuracy_score(y_test, y_pred_cart)
print(f'Accuracy: {accuracy_cart:.2f}')

# Matriz de confusión
conf_matrix_cart = confusion_matrix(y_test, y_pred_cart)
print('Matriz de confusión:')
print(conf_matrix_cart)

# Reporte de clasificación
class_report_CART = classification_report(y_test, y_pred_cart)
print('Classification Report:')
print(class_report_CART)

# Crear Matriz
sns.heatmap(conf_matrix_cart, annot=True, fmt='d', cmap='Blues', cbar=False)

# Configurar las etiquetas de los ejes y el título
plt.xlabel('Predicciones')
plt.ylabel('Realidad')
plt.title('Matriz de confusión sin busqueda de hiperparametros')

# Mostrar la visualización
plt.show()
```

```
Accuracy: 0.80
Matriz de confusión:
[[62  8]
 [13 23]]
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.89      0.86        70
           1       0.74      0.64      0.69        36

    accuracy                           0.80       106
   macro avg       0.78      0.76      0.77       106
weighted avg       0.80      0.80      0.80       106
```

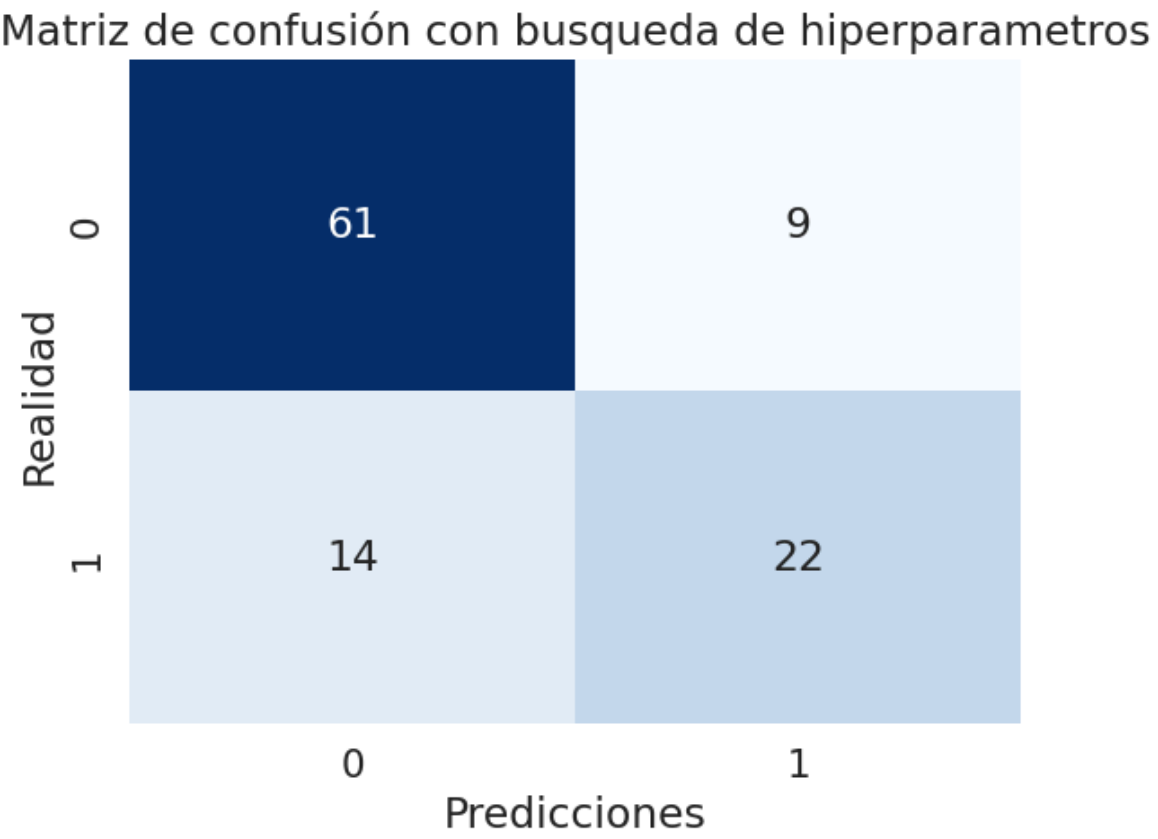


Arboles de decisión con criterio CART con busqueda de hiperparametros

In [ ]:
```
# Definir los hiperparámetros a ajustar
param_grid = {
    'max_depth': [None, 5, 10, 15],  # Vary the maximum depth of the tree
    'min_samples_split': [2, 5, 10],  # Vary the minimum number of samples req
    'min_samples_leaf': [1, 2, 4]  # Vary the minimum number of samples requir
}

# Definir el protocolo de evaluación K-fold
kfold = KFold(n_splits=5, shuffle=True, random_state=42)

# Inicializar el GridSearchCV
grid_search_cart = GridSearchCV(estimator=DecisionTreeClassifier(criterion='gi
                           param_grid=param_grid,
                           cv=kfold,
                           scoring='accuracy',
                           verbose=1,
                           n_jobs=-1)

# Entrenar el modelo utilizando GridSearchCV
grid_search_cart.fit(X_train, y_train)

# Mejores hiperparámetros encontrados
best_params_cart = grid_search_cart.best_params_
print("Best Parameters:", best_params_cart)

# Mejor modelo
best_model_cart = grid_search_cart.best_estimator_

y_pred_cart_2 = best_model_cart.predict(X_test)


# Calcular la precisión
accuracy_cart_2 = accuracy_score(y_test, y_pred_cart_2)
print(f'Accuracy: {accuracy_cart_2:.2f}')

# Matriz de confusión
conf_matrix_cart_2 = confusion_matrix(y_test, y_pred_cart_2)
print('Matriz de confusión:')
print(conf_matrix_cart_2)

# Reporte de clasificación
class_report_cart_2 = classification_report(y_test, y_pred_cart_2)
print('Classification Report:')
print(class_report_cart_2)

# Crear Matriz
sns.heatmap(conf_matrix_cart_2, annot=True, fmt='d', cmap='Blues', cbar=False)

# Configurar las etiquetas de los ejes y el título
plt.xlabel('Predicciones')
plt.ylabel('Realidad')
plt.title('Matriz de confusión con busqueda de hiperparametros')

# Mostrar la visualización
```

```
plt.show()
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
Best Parameters: {'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split':
2}
Accuracy: 0.78
Matriz de confusión:
[[61  9]
 [14 22]]
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.87      0.84        70
           1       0.71      0.61      0.66        36

    accuracy                           0.78       106
   macro avg       0.76      0.74      0.75       106
weighted avg       0.78      0.78      0.78       106
```



###Conclusión:

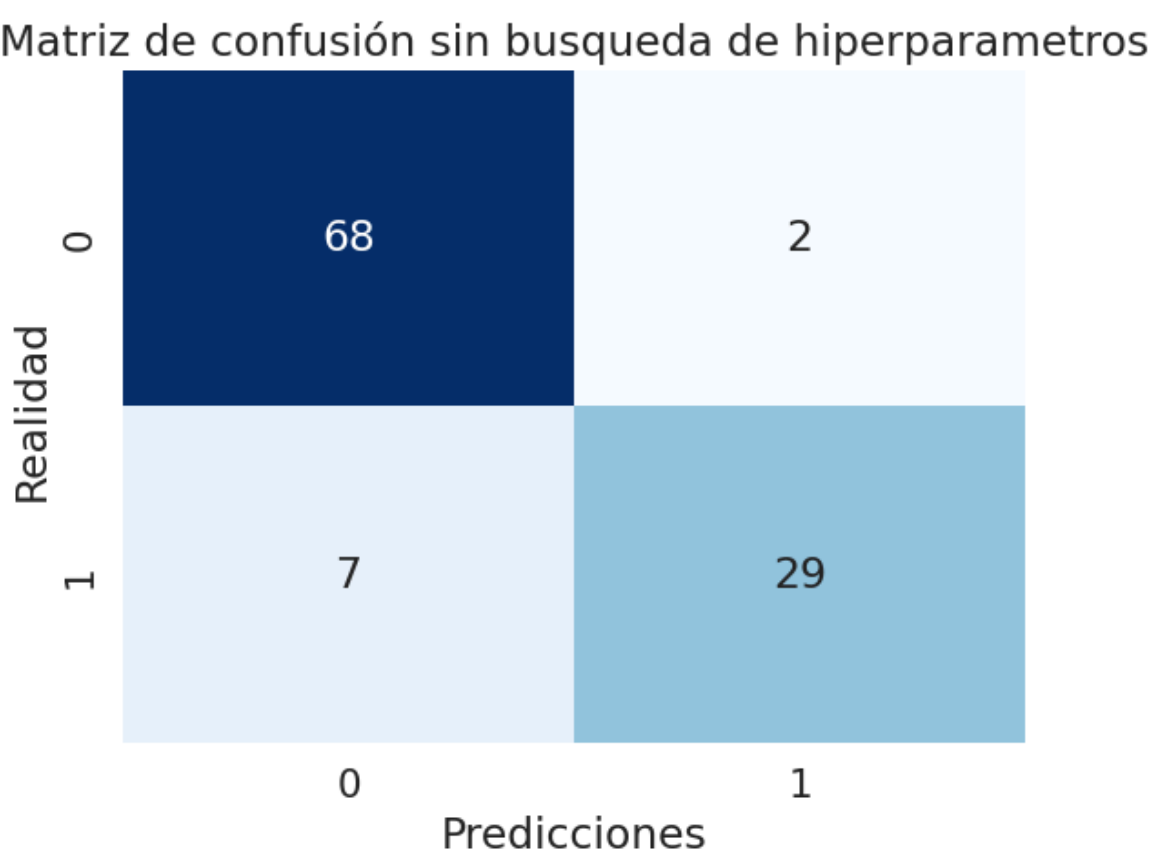
El modelo de arboles de decisión con CART de entropía y el modelo con búsqueda de hiperparámetros, estós arrojaron como resultado de accuracy 82% y 83%, y tiene un recall de 86% y 81% respectivamente mostrando que el segundo modelo obtuvo una precisión ligeramente superior, el primer modelo demostró ser más robusto en términos de recall, lo que puede ser crucial en situaciones donde la identificación precisa de instancias positivas es de alta.

Modelo 6: Bagging

Bagging sin busqueda de hiperparametros

In [ ]:
```
from sklearn.ensemble import BaggingClassifier

# Modelo Baggin (usando como base estimador de gini)
# Entrenar el modelo Bagging con un árbol de decisión
base_estimator = DecisionTreeClassifier(criterion='gini', random_state=42)
bagging = BaggingClassifier(estimator=base_estimator, n_estimators=100, random
bagging.fit(X_train, y_train)

# Evaluar el modelo
y_pred_bg = bagging.predict(X_test)

# Calcular la precisión
accuracy_bg = accuracy_score(y_test, y_pred_bg)
print(f'Accuracy: {accuracy_bg:.2f}')

# Matriz de confusión
conf_matrix_bg = confusion_matrix(y_test, y_pred_bg)
print('Confusion Matrix:')
print(conf_matrix_bg)

# Reporte de clasificación
class_report_Bag = classification_report(y_test, y_pred_bg)
print('Classification Report:')
print(class_report_Bag)

# Crear Matriz
sns.heatmap(conf_matrix_bg, annot=True, fmt='d', cmap='Blues', cbar=False)

# Configurar las etiquetas de los ejes y el título
plt.xlabel('Predicciones')
plt.ylabel('Realidad')
plt.title('Matriz de confusión sin busqueda de hiperparametros')

# Mostrar la visualización
plt.show()
```

```
Accuracy: 0.92
Confusion Matrix:
[[68  2]
 [ 7 29]]
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.97      0.94        70
           1       0.94      0.81      0.87        36

    accuracy                           0.92       106
   macro avg       0.92      0.89      0.90       106
weighted avg       0.92      0.92      0.91       106
```

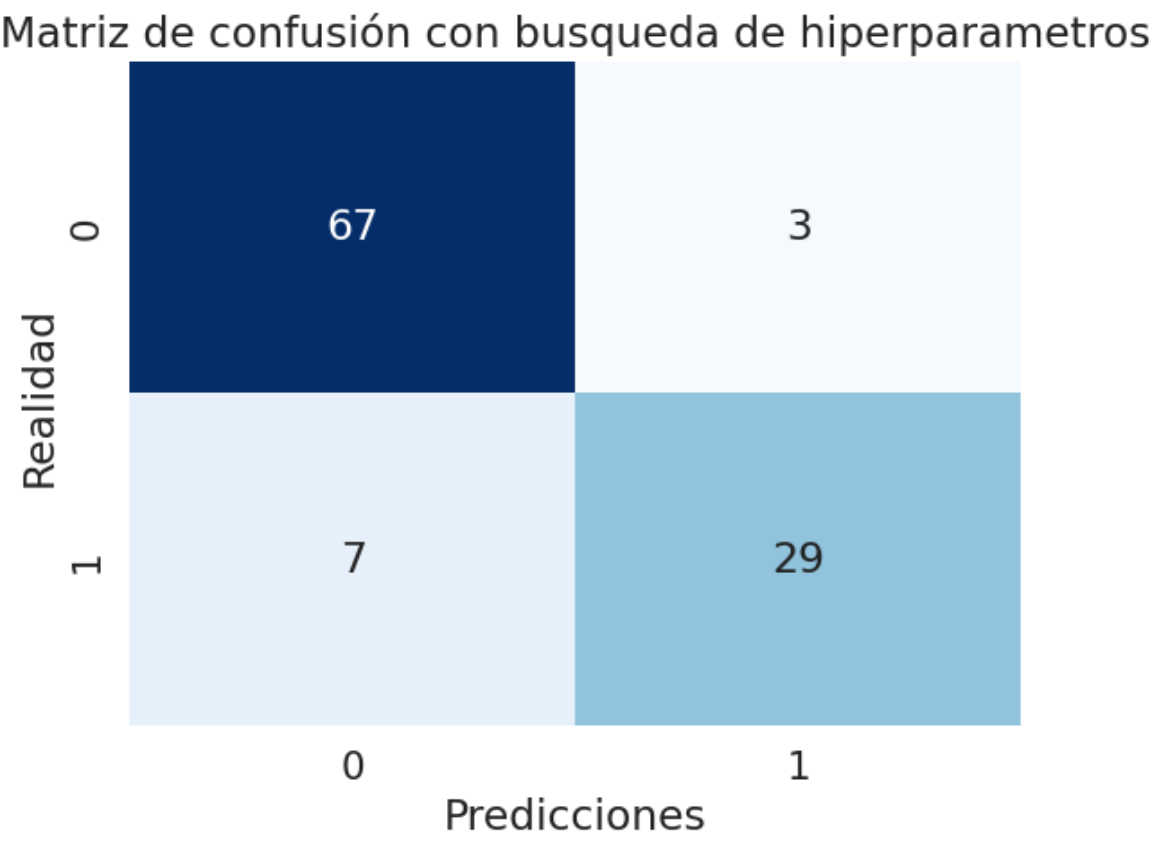


Bagging con busqueda de hiperparametros

In [ ]:
```
# Definir los hiperparámetros para el árbol de decisión
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Definir el protocolo de evaluación K-fold
kfold = KFold(n_splits=5, shuffle=True, random_state=42)

# Inicializar GridSearchCV para el árbol de decisión
grid_search_dt = GridSearchCV(estimator=DecisionTreeClassifier(random_state=42
                              param_grid=param_grid,
                              cv=kfold,
                              scoring='accuracy',
                              verbose=1,
                              n_jobs=-1)

# Entrenar el árbol de decisión utilizando GridSearchCV
grid_search_dt.fit(X_train, y_train)

# Mejores hiperparámetros encontrados para el árbol de decisión
best_params_dt = grid_search_dt.best_params_
print("Best Parameters for Decision Tree:", best_params_dt)

# Construir el modelo de Bagging utilizando el árbol de decisión con los hiper
base_estimator_dt = DecisionTreeClassifier(random_state=42, **best_params_dt)
bagging_2 = BaggingClassifier(estimator=base_estimator_dt, n_estimators=100, r

# Entrenar el modelo de Bagging
bagging_2.fit(X_train, y_train)

# Evaluar el modelo de Bagging
y_pred_bg_2 = bagging_2.predict(X_test)

# Calcular la precisión
accuracy_bg_2 = accuracy_score(y_test, y_pred_bg_2)
print(f'Accuracy: {accuracy_bg_2:.2f}')

# Matriz de confusión
conf_matrix_bg_2 = confusion_matrix(y_test, y_pred_bg_2)
print('Confusion Matrix:')
print(conf_matrix_bg_2)

# Reporte de clasificación
class_report_Bag_2 = classification_report(y_test,  y_pred_bg_2)
print('Classification Report:')
print(class_report_Bag_2)

# Matriz de confusión
conf_matrix_bg_2 = confusion_matrix(y_test, y_pred_bg_2)
print('Confusion Matrix:')
print(conf_matrix_bg_2)

# Crear Matriz
sns.heatmap(conf_matrix_bg_2, annot=True, fmt='d', cmap='Blues', cbar=False)
```

```
# Configurar las etiquetas de los ejes y el título
plt.xlabel('Predicciones')
plt.ylabel('Realidad')
plt.title('Matriz de confusión con busqueda de hiperparametros')

# Mostrar la visualización
plt.show()
```

```
Fitting 5 folds for each of 72 candidates, totalling 360 fits
Best Parameters for Decision Tree: {'criterion': 'gini', 'max_depth': 5, 'min
_samples_leaf': 1, 'min_samples_split': 2}
Accuracy: 0.91
Confusion Matrix:
[[67  3]
 [ 7 29]]
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.96      0.93        70
           1       0.91      0.81      0.85        36

    accuracy                           0.91       106
   macro avg       0.91      0.88      0.89       106
weighted avg       0.91      0.91      0.90       106

Confusion Matrix:
[[67  3]
 [ 7 29]]
```



###Conclusión:

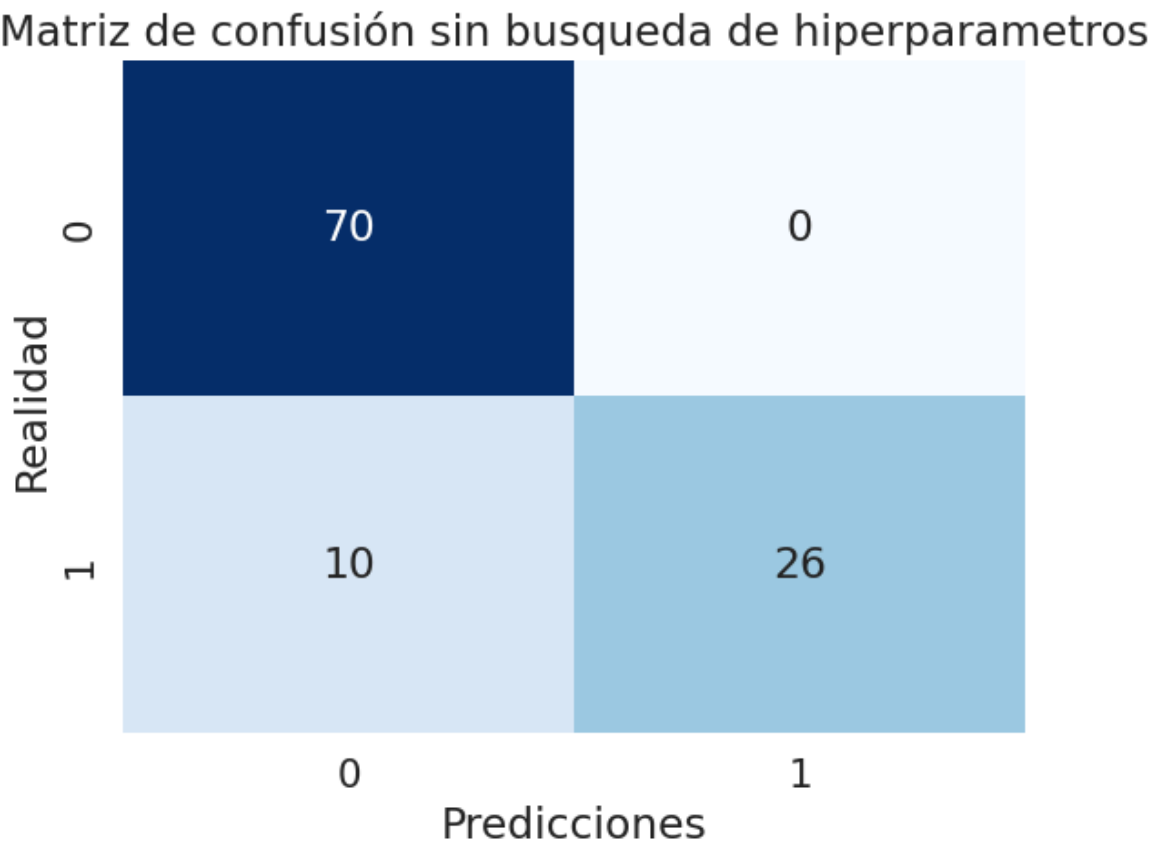
Se corre el modelo de Bagging y el modelo con búsqueda de hiperparámetros arrojaron como resultado de accuracy 87%, y tiene un recall de 91% y 92% respectivamente, evidenciando que el segundo modelo demostró una capacidad ligeramente superior para identificar correctamente las instancias positivas en comparación con el primero.

Modelo 7: Random Forest

Random Forest sin busqueda de hiperparametros

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier

#Modelo Random Forest

# Entrenar el modelo Random Forest
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

# Evaluar el modelo
y_pred_rf = rf.predict(X_test)

# Calcular la precisión
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print(f'Accuracy: {accuracy_rf:.2f}')

# Matriz de confusión
conf_matrix_rf = confusion_matrix(y_test, y_pred_rf)
print('Confusion Matrix:')
print(conf_matrix_rf)

# Reporte de clasificación
class_report_RF = classification_report(y_test, y_pred_rf)
print('Classification Report:')
print(class_report_RF)

# Crear Matriz
sns.heatmap(conf_matrix_rf, annot=True, fmt='d', cmap='Blues', cbar=False)

# Configurar las etiquetas de los ejes y el título
plt.xlabel('Predicciones')
plt.ylabel('Realidad')
plt.title('Matriz de confusión sin busqueda de hiperparametros')

# Mostrar la visualización
plt.show()
```

```
Accuracy: 0.91
Confusion Matrix:
[[70  0]
 [10 26]]
Classification Report:
              precision    recall  f1-score   support

           0       0.88      1.00      0.93        70
           1       1.00      0.72      0.84        36

    accuracy                           0.91       106
   macro avg       0.94      0.86      0.89       106
weighted avg       0.92      0.91      0.90       106
```

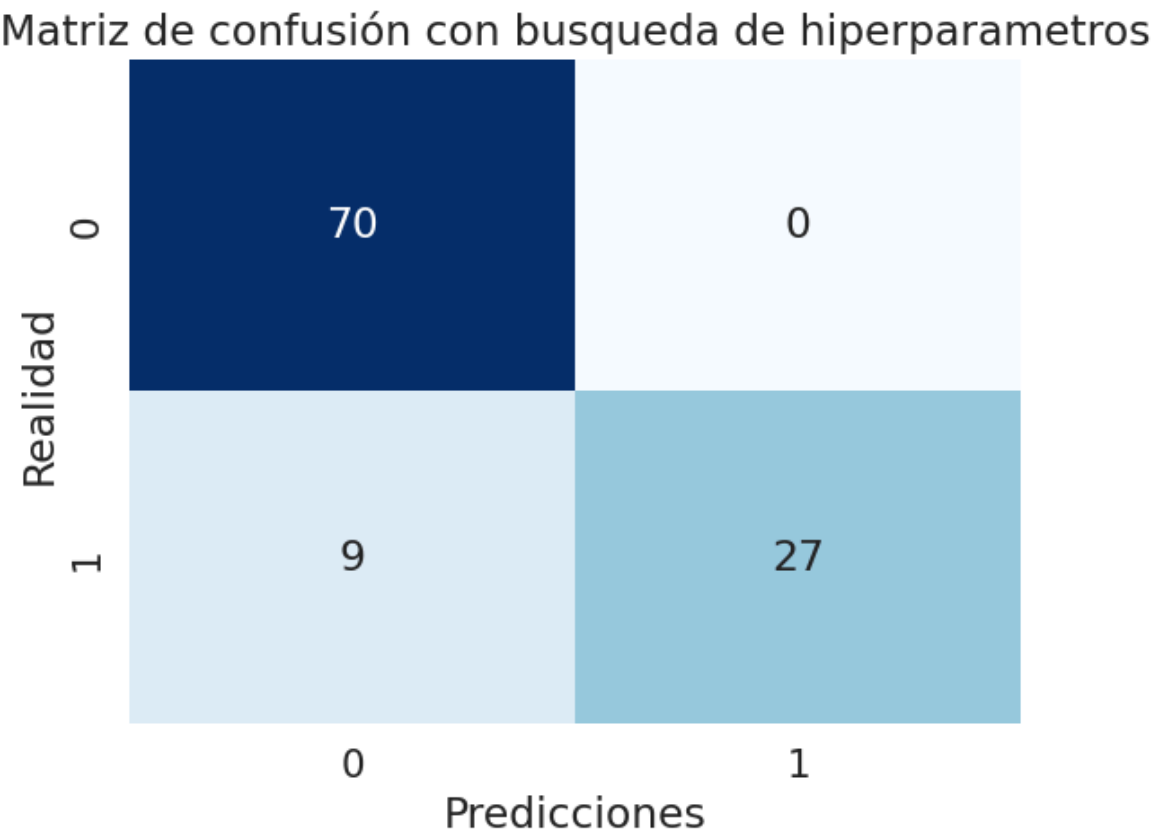


Random Forest con busqueda de hiperparametros

In [ ]:
```
# Definir los hiperparámetros a ajustar
param_grid = {
'n_estimators': [50, 100, 150],
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Definir el protocolo de evaluación K-fold
kfold = KFold(n_splits=5, shuffle=True, random_state=42)

# Inicializar GridSearchCV para el Random Forest
grid_search_rf = GridSearchCV(estimator=RandomForestClassifier(random_state=42
                              param_grid=param_grid,
                              cv=kfold,
                              scoring='accuracy',
                              verbose=1,
                              n_jobs=-1)

# Entrenar el modelo utilizando GridSearchCV
grid_search_rf.fit(X_train, y_train)

# Mejores hiperparámetros encontrados
best_params_rf = grid_search_rf.best_params_
print("Best Parameters for Random Forest:", best_params_rf)

# Mejor modelo
best_model_rf = grid_search_rf.best_estimator_

# Evaluar el modelo en los datos de prueba
y_pred_rf_2 = best_model_rf.predict(X_test)

# Calcular la precisión
accuracy_rf_2 = accuracy_score(y_test, y_pred_rf_2)
print(f'Accuracy: {accuracy_rf_2:.2f}')

# Matriz de confusión
conf_matrix_rf_2 = confusion_matrix(y_test, y_pred_rf_2)
print('Confusion Matrix:')
print(conf_matrix_rf_2)

# Reporte de clasificación
class_report_RF_2 = classification_report(y_test, y_pred_rf_2)
print('Classification Report:')
print(class_report_RF_2)

# Crear Matriz
sns.heatmap(conf_matrix_rf_2, annot=True, fmt='d', cmap='Blues', cbar=False)

# Configurar las etiquetas de los ejes y el título
plt.xlabel('Predicciones')
plt.ylabel('Realidad')
plt.title('Matriz de confusión con busqueda de hiperparametros')

# Mostrar la visualización
```

```
plt.show()
```

```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
Best Parameters for Random Forest: {'max_depth': 10, 'min_samples_leaf': 1,
'min_samples_split': 2, 'n_estimators': 150}
Accuracy: 0.92
Confusion Matrix:
[[70  0]
 [ 9 27]]
Classification Report:
              precision    recall  f1-score   support

           0       0.89      1.00      0.94        70
           1       1.00      0.75      0.86        36

    accuracy                           0.92       106
   macro avg       0.94      0.88      0.90       106
weighted avg       0.92      0.92      0.91       106
```



###Conclusión:

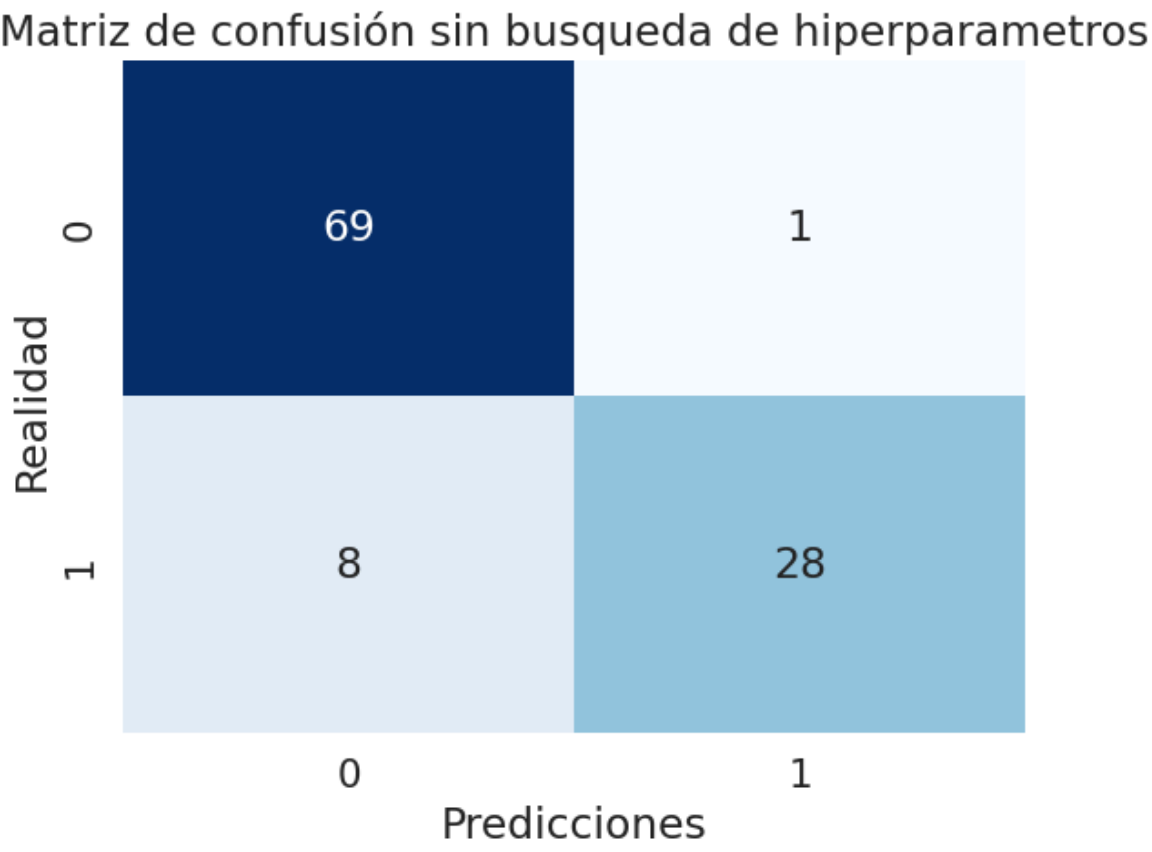
En los modelos de Random Forest y el modelo con búsqueda de hiperparámetros los modelos arrojaron como resultado de accuracy 89% y 91%, y tiene un recall de 94% y 96% respectivamente, evidenciando que el segundo modelo demostró una capacidad superior para identificar correctamente las instancias positivas en comparación con el primero.

Modelo 8: XGBoost

XGBoost sin busqueda de hiperparametros

In [ ]:
```
import xgboost as xgb

# XGBoost

# Entrenar el modelo XGBoost
xgb_model = xgb.XGBClassifier(objective="binary:logistic", random_state=42)
xgb_model.fit(X_train, y_train)

# Evaluar el modelo
y_pred_xg = xgb_model.predict(X_test)

# Calcular la precisión
accuracy_xg = accuracy_score(y_test, y_pred_xg)
print(f'Accuracy: {accuracy_xg:.2f}')

# Matriz de confusión
conf_matrix_xg = confusion_matrix(y_test, y_pred_xg)
print('Confusion Matrix:')
print(conf_matrix_xg)

# Reporte de clasificación
class_report_XGB = classification_report(y_test, y_pred_xg)
print('Classification Report:')
print(class_report_XGB)

# Crear Matriz
sns.heatmap(conf_matrix_xg, annot=True, fmt='d', cmap='Blues', cbar=False)

# Configurar las etiquetas de los ejes y el título
plt.xlabel('Predicciones')
plt.ylabel('Realidad')
plt.title('Matriz de confusión sin busqueda de hiperparametros')

# Mostrar la visualización
plt.show()
```

```
Accuracy: 0.92
Confusion Matrix:
[[69  1]
 [ 8 28]]
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.99      0.94        70
           1       0.97      0.78      0.86        36

    accuracy                           0.92       106
   macro avg       0.93      0.88      0.90       106
weighted avg       0.92      0.92      0.91       106
```

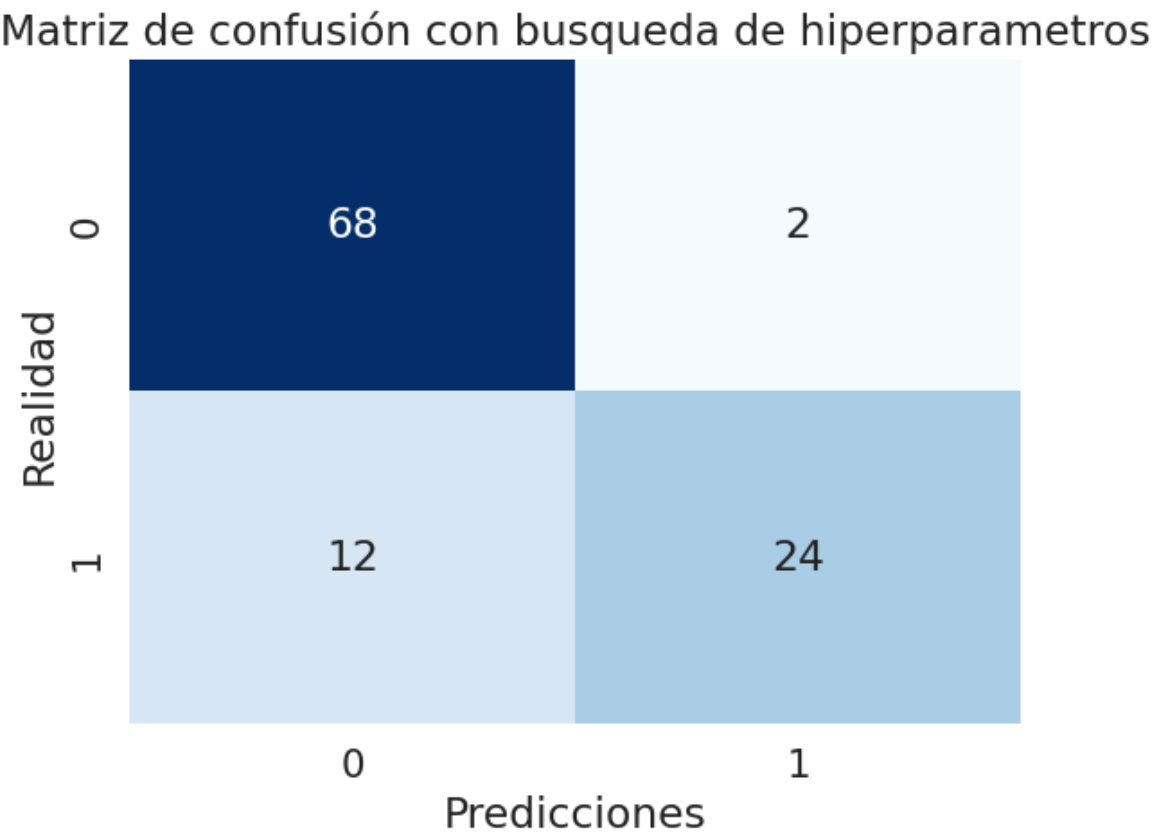


XGBoost con busqueda de hiperparametros

In [ ]:
```
# Definir los hiperparámetros a ajustar
param_grid = {
    'learning_rate': [0.01, 0.05, 0.1, 0.3, 0.5, 0.7, 1.0],
    'max_depth': [3, 5, 7, 9, 11],
    'n_estimators': [50, 100, 150, 200, 250],
}

# Definir el protocolo de evaluación K-fold
kfold = KFold(n_splits=5, shuffle=True, random_state=42)

# Inicializar GridSearchCV para XGBoost
grid_search_xgb = GridSearchCV(estimator=xgb.XGBClassifier(objective="binary:l
                                param_grid=param_grid,
                                cv=kfold,
                                scoring='accuracy',
                                verbose=1,
                                n_jobs=-1)

# Entrenar el modelo utilizando GridSearchCV
grid_search_xgb.fit(X_train, y_train)

# Mejores hiperparámetros encontrados
best_params_xgb = grid_search_xgb.best_params_
print("Best Parameters for XGBoost:", best_params_xgb)

# Mejor modelo
best_model_xgb = grid_search_xgb.best_estimator_

# Evaluar el modelo en los datos de prueba
y_pred_xgb = best_model_xgb.predict(X_test)

# Calcular la precisión
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)
print(f'Accuracy: {accuracy_xgb:.2f}')

# Matriz de confusión
conf_matrix_xgb = confusion_matrix(y_test, y_pred_xgb)
print('Confusion Matrix:')
print(conf_matrix_xgb)

# Reporte de clasificación
class_report_XGB = classification_report(y_test, y_pred_xgb)
print('Classification Report:')
print(class_report_XGB)

# Crear Matriz
sns.heatmap(conf_matrix_xgb, annot=True, fmt='d', cmap='Blues', cbar=False)

# Configurar las etiquetas de los ejes y el título
plt.xlabel('Predicciones')
plt.ylabel('Realidad')
plt.title('Matriz de confusión con busqueda de hiperparametros')

# Mostrar la visualización
```

```
plt.show()
```

```
Fitting 5 folds for each of 175 candidates, totalling 875 fits
Best Parameters for XGBoost: {'learning_rate': 1.0, 'max_depth': 9, 'n_estima
tors': 100}
Accuracy: 0.87
Confusion Matrix:
[[68  2]
 [12 24]]
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.97      0.91        70
           1       0.92      0.67      0.77        36

    accuracy                           0.87       106
   macro avg       0.89      0.82      0.84       106
weighted avg       0.87      0.87      0.86       106
```



###Conclusión:

Al correr el modelo de modelos de XGBoost y el modelo con búsqueda de hiperparámetros los modelos arrojaron como resultado de accuracy 86% y 85%, y tiene un recall de 91% y 91% respectivamente, En este caso específico, la búsqueda de hiperparámetros no genero un impacto importante en el rendimiento del modelo en términos de recall, aunque hubo una ligera diferencia en el accuracy. Esto indica que los hiperparámetros predeterminados en el modelo sin búsqueda de hiperparámetros fueron lo suficientemente buenos como para lograr un rendimiento comparable al modelo optimizado

###RESUMEN DE RESULTADOS DE LOS MODELOS USADOS

In [ ]:
```
# Lista de nombres de las predicciones sin búsqueda y con búsqueda
predicciones = {
    "Sin busqueda": [y_pred_rl, y_pred_knn, y_pred_nb, y_pred_id3, y_pred_cart
    "Con busqueda": [y_pred_cv, y_pred_cv_knn, None, y_pred_id3_2, y_pred_cart
}

nombres_modelos = [
    "Regresion logistica", "KNN", "Bayes", "Arbol de desicion ID3",
    "Arbol de desicion Cart", "Bagging", "Random forest", "XGBoost"
]

# Inicializar diccionario para almacenar los resultados
resultados = {
    "Modelo": nombres_modelos,
    "Accuracy (Sin busqueda)": [],
    "Recall (Sin busqueda)": [],
    "Accuracy (Con busqueda)": [],
    "Recall (Con busqueda)": []
}

# Calcular las métricas
for i, modelo in enumerate(nombres_modelos):
    # Sin busqueda
    y_pred_sin = predicciones["Sin busqueda"][i]
    if y_pred_sin is not None:
        resultados["Accuracy (Sin busqueda)"].append(accuracy_score(y_test, y_
        resultados["Recall (Sin busqueda)"].append(recall_score(y_test, y_pred
    else:
        resultados["Accuracy (Sin busqueda)"].append(None)
        resultados["Recall (Sin busqueda)"].append(None)

    # Con busqueda
    y_pred_con = predicciones["Con busqueda"][i]
    if y_pred_con is not None:
        resultados["Accuracy (Con busqueda)"].append(accuracy_score(y_test, y_
        resultados["Recall (Con busqueda)"].append(recall_score(y_test, y_pred
    else:
        resultados["Accuracy (Con busqueda)"].append(None)
        resultados["Recall (Con busqueda)"].append(None)

# Crear un DataFrame con los resultados
df_resultados = pd.DataFrame(resultados)


df_resultados
```

Out[121]:

	Modelo	Accuracy (Sin busqueda)	Recall (Sin busqueda)	Accuracy (Con busqueda)	Recall (Con busqueda)
0	Regresion logistica	0.896226	0.928571	0.858491	0.957143
1	KNN	0.849057	0.985714	0.811321	0.942857
2	Bayes	0.839623	0.914286	NaN	NaN
3	Arbol de desicion ID3	0.811321	0.914286	0.783019	0.885714
4	Arbol de desicion Cart	0.801887	0.885714	0.783019	0.871429
5	Bagging	0.915094	0.971429	0.905660	0.957143
6	Random forest	0.905660	1.000000	0.915094	1.000000
7	XGBoost	0.915094	0.985714	0.867925	0.971429

Selección de Modelo para la Predicción del Síndrome de Ovario Poliquístico

Después de evaluar varios modelos de aprendizaje automático, hemos determinado que el modelo de **Random Forest con búsqueda de hiperparámetros** es el más adecuado para predecir el síndrome de ovario poliquístico. Este modelo ha mostrado un rendimiento superior en comparación con los demás modelos evaluados, alcanzando una **precisión (accuracy) de 0.91** y una **recuperación (recall) de 0.96**. Estas métricas indican que el modelo no solo es capaz de predecir correctamente una alta proporción de los casos, sino que también tiene una excelente capacidad para identificar todos los casos positivos de la enfermedad.

En resumen, la combinación de alta precisión y alta recuperación obtenida mediante la optimización de hiperparámetros hace que el modelo de Random Forest sea nuestra mejor opción para esta tarea. Este rendimiento robusto sugiere que el modelo es confiable y eficiente para el diagnóstico del síndrome de ovario poliquístico, lo que puede contribuir significativamente a la mejora en la identificación y el tratamiento oportuno de esta condición.

In [ ]:
```
print("Los mejores hiperparametros para el modelo seleccionado fueron los sigu
```

```
Los mejores hiperparametros para el modelo seleccionado fueron los siguiente
s: {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estima
tors': 150}
```

COMPONENTES DE ANÁLISIS PRINCIPALES (PCA)

El Análisis de Componentes Principales (PCA, por sus siglas en inglés) es una técnica de reducción de dimensionalidad ampliamente utilizada en la ciencia de datos y el análisis de datos. Su objetivo principal es transformar un conjunto de variables posiblemente

correlacionadas en un conjunto más pequeño de variables no correlacionadas, denominadas

El dataset de PCOS contiene muchas variables. PCA reduce el número de variables al encontrar nuevas variables no correlacionadas (componentes principales) que son combinaciones lineales de las originales y que explican la mayor parte de la varianza en los datos.

Escalado de Datos

Primero, escalamos los datos utilizando StandardScaler de scikit-learn. Esto asegura que todas las características tengan una media de 0 y una desviación estándar de 1, lo cual es importante para PCA, ya que es sensible a las escalas de las variables.

In [ ]:
```
# Escalar los datos
scaler = StandardScaler(with_mean=True, with_std=True)
scaler.fit(df)

# Imprimir la media de cada característica
print(scaler.mean_)
```

```
[3.34600760e-01 3.14182510e+01 5.96754753e+01 1.56381749e+02
 2.43564639e+01 1.38098859e+01 7.34619772e+01 1.92566540e+01
 1.11577567e+01 2.81368821e-01 4.93916350e+00 7.67587454e+00
 3.85931559e-01 2.85171103e-01 6.81005620e+02 2.44794627e+02
 1.48321160e+01 6.55696198e+00 7.00760456e+00 3.79828897e+01
 3.38498099e+01 8.92110266e-01 2.99848289e+00 5.67236365e+00
 2.45695437e+01 5.06406616e+01 6.19412548e-01 9.98235741e+01
 3.85931559e-01 2.77566540e-01 3.09885932e-01 4.60076046e-01
 5.00000000e-01 5.19011407e-01 2.49049430e-01 1.14885932e+02
 7.71102662e+01 6.24334601e+00 6.72813688e+00 1.52353612e+01
 1.54991825e+01 8.52693916e+00]
```

In [ ]:
```
# Transformar los datos escalados
df_std = scaler.transform(df)
```

Segundo, aplicaremos el Análisis de Componentes Principales (PCA) a los datos estandarizados. Utilizaremos la clase PCA de scikit-learn para ajustar el modelo PCA y transformar los datos.

In [ ]:
```
from sklearn.decomposition import PCA

# Aplicamos PCA
pca = PCA()
df_proyectado = pca.fit_transform(df_std)
```

Revisamos los componentes principales obtenidos del PCA. Los componentes principales son vectores que describen la dirección en el espacio original de las características en la que se encuentra la mayor varianza de los datos.

In [ ]:
```
# Obtener los componentes principales
print(pca.components_)
```

```
[[ 3.36168029e-01 -6.59403454e-02  3.04971693e-01 ...  2.17102210e-02
   1.15019278e-02  4.72625821e-02]
 [-2.34900133e-01  2.03333590e-01  3.31789749e-01 ... -1.82261594e-01
  -2.03025442e-01 -8.90798351e-02]
 [-4.48501724e-02 -1.07030305e-01  7.11795069e-02 ...  1.44490863e-01
   1.58457781e-01 -5.53983922e-02]
 ...
 [ 3.89819452e-03 -3.47968091e-02 -2.82918175e-02 ...  2.10738112e-04
  -3.29822740e-03  5.37524051e-03]
 [ 2.78256305e-03  6.87116464e-03 -7.04675036e-01 ...  1.13159995e-03
   2.10732660e-03  1.92165670e-03]
 [-4.47274952e-03  5.74119294e-03  1.74268418e-02 ... -3.44934158e-04
   4.87036534e-04 -1.52588364e-03]]
```

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

Out[127]:
```
Index(['pcos(s-n)', 'edad-a', 'peso-kg', 'estatura-cm', 'imc',
       'grupo-sanguineo', 'frecuencia-cardiaca-bpm',
       'frecuencia-respiratoria-respiraciones/min', 'hemoglobina-g/dl',
       'ciclo-r', 'duracion-ciclo-d', 'tiempo-casada-a', 'embarazada(s-n)',
       'nro-abortos', 'h-beta-hcg-I-mIU/mL', 'h-beta-hcg-II-mIU/mL',
       'h-fsh-mIU/mL', 'h-lh-mIU/mL', 'h-fsh/h-lh', 'cadera-pulg',
       'cintura-pulg', 'ind-cintura/cadera', 'h-tsh-mIU/L', 'h-amh-ng/mL',
       'h-prl-ng/mL', 'ex-vit-d3-ng/mL', 'h-prg-ng/mL', 'ex-rbs-mg/dl',
       'ganancia-peso(s-n)', 'crecimiento-cabello(s-n)',
       'oscurecimiento-piel(s-n)', 'perdida-cabello(s-n)',
       'barro-espinilla(s-n)', 'comida-rapida(s-n)', 'ejercicio-regular(s-
n)',
       'ps-sistolica-mmHg', 'ps-diastolica-mmHg', 'nro-foliculos-ovario-izq',
       'nro-foliculos-ovario-der', 'prom-tam-foliculos-ovario-izq-mm',
       'prom-tam-foliculos-ovario-der-mm', 'endometrio-mm'],
      dtype='object')
```

In [ ]:
```
print(pca.explained_variance_ratio_)
```

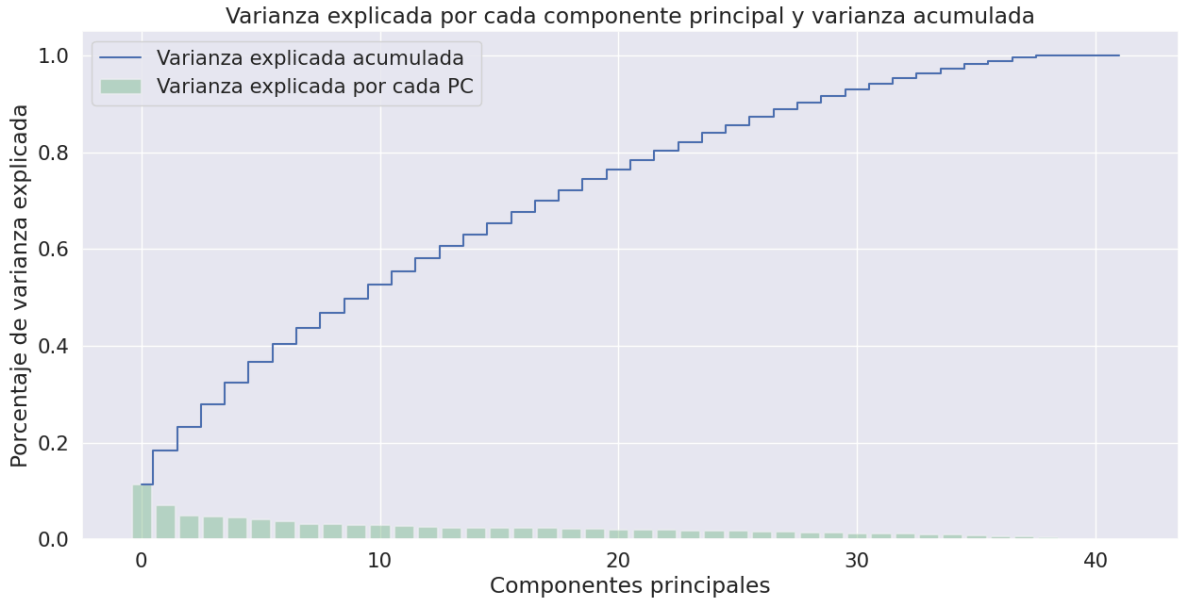
```
[1.12571655e-01 7.07414113e-02 4.90732367e-02 4.74197031e-02
 4.45672259e-02 4.20884401e-02 3.80902021e-02 3.25024173e-02
 3.07828836e-02 2.95382026e-02 2.91511315e-02 2.75286187e-02
 2.67743162e-02 2.47486354e-02 2.40711438e-02 2.37055901e-02
 2.36106569e-02 2.29544517e-02 2.21908211e-02 2.17094144e-02
 2.03814394e-02 2.00085546e-02 1.93877281e-02 1.78884867e-02
 1.76471967e-02 1.75074308e-02 1.64917020e-02 1.59164892e-02
 1.41871978e-02 1.35610347e-02 1.26466652e-02 1.20900013e-02
 1.16288413e-02 1.02191725e-02 9.44838537e-03 9.09608596e-03
 7.05221416e-03 6.47358844e-03 3.84542071e-03 6.13615270e-04
 5.77002879e-05 3.08920502e-05]
```

In [ ]:
```
np.sum(pca.explained_variance_ratio_[0:28])
```

Out[129]:
```
0.8890491850900242
```

A pesar de tener muchas características (42), los primeros pocos componentes principales pueden capturar una proporción significativa de la varianza. Por ejemplo, con solo los primeros 28 componentes, capturamos alrededor del 88.9% de la varianza.

In [ ]:
```
# Varianza explicada por cada componente principal
var_exp = pca.explained_variance_ratio_

# Varianza acumulada
cum_var_exp = np.cumsum(var_exp)

# Visualización
plt.figure(figsize=(15, 7))
plt.bar(range(len(var_exp)), var_exp, alpha=0.3333, align='center', label='Var
plt.step(range(len(cum_var_exp)), cum_var_exp, where='mid', label='Varianza ex
plt.ylabel('Porcentaje de varianza explicada')
plt.xlabel('Componentes principales')
plt.legend(loc='best')
plt.title('Varianza explicada por cada componente principal y varianza acumula
plt.show()
```



Conclusiones PCA

- El PCA ha permitido reducir la complejidad del dataset original, compuesto por 42 características, a un número menor de componentes principales que capturan la mayoría de la variabilidad en los datos. Esta reducción facilita la interpretación y el análisis posterior.
- Los primeros cinco componentes principales en conjunto explican aproximadamente el 31.82% de la varianza total. Para explicar más del 50% de la varianza total, se necesitan aproximadamente los primeros 16 componentes principales.
- La varianza acumulada muestra que poco más de los primeros 20 componentes principales son necesarios para capturar aproximadamente el 80% de la varianza total. Esto indica que, aunque se puede reducir la dimensionalidad del dataset, se necesita un

número considerable de componentes para mantener una cantidad significativa de información.

Clustering

In [ ]:
```
df.describe()
```

Out[137]:

	pcos(s-n)	edad-a	peso-kg	estatura-cm	imc	grupo-sanguineo	frecuencia-cardiaca-bpm	resp
count	526.000000	526.000000	526.000000	526.000000	526.000000	526.000000	526.000000	
mean	0.334601	31.418251	59.675475	156.381749	24.356464	13.809886	73.461977	
std	0.472300	5.417781	10.950673	5.989715	4.034686	1.843903	2.697703	
min	0.000000	20.000000	31.000000	137.000000	12.400000	11.000000	70.000000	
25%	0.000000	27.000000	52.000000	152.000000	21.900000	13.000000	72.000000	
50%	0.000000	31.000000	59.800000	156.000000	24.300000	14.500000	72.000000	
75%	1.000000	35.000000	65.000000	160.000000	26.700000	15.000000	74.000000	
max	1.000000	48.000000	108.000000	180.000000	38.900000	18.000000	82.000000	

8 rows × 42 columns

K-MEANS

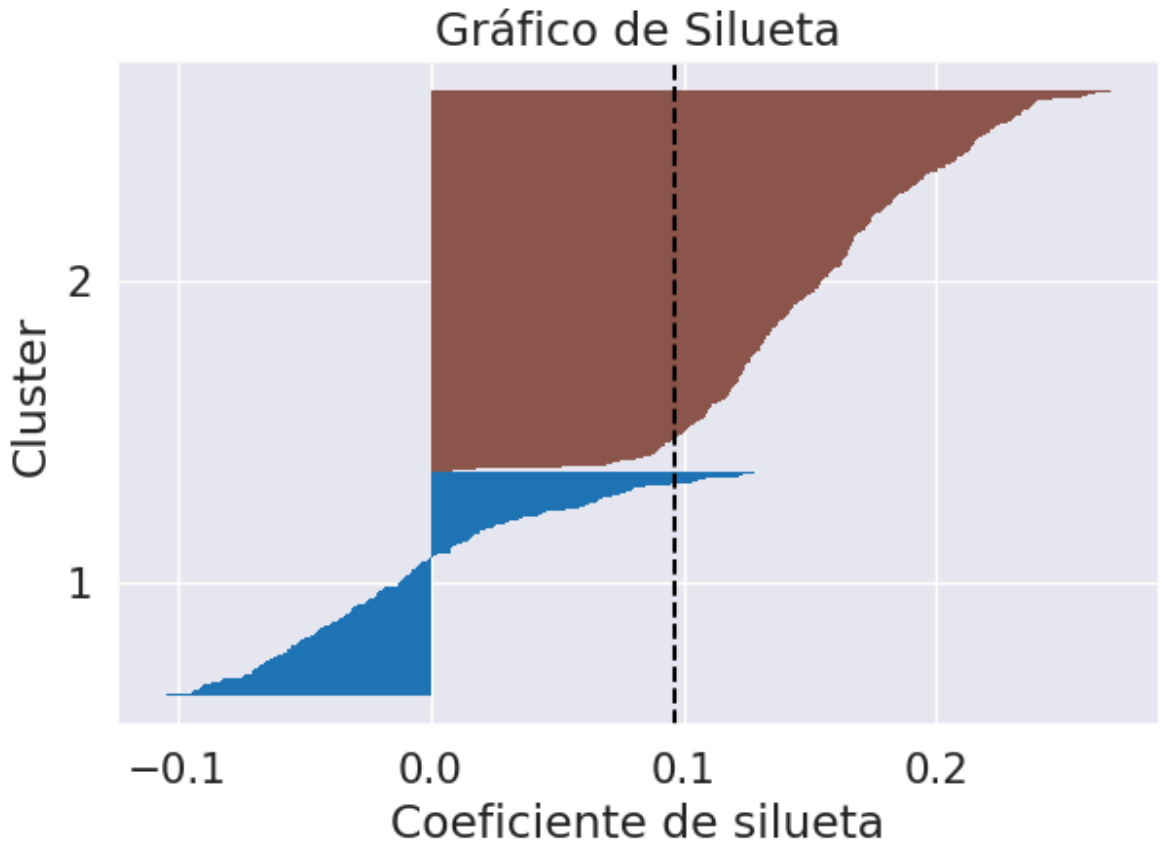
In [ ]:
```
from sklearn.cluster import KMeans

# Filtrar las variables eliminando las dicotómicas
variables_continuas = [var for var in df if '(s-n)' not in var]

# Crear un nuevo DataFrame con solo las variables numéricas continuas
df_continuo = df[variables_continuas]

# Escalar los datos
scaler = StandardScaler()
df_std = scaler.fit_transform(df_continuo)

# Inicializar K-Means
kmeans = KMeans(n_clusters=4, random_state=0)

# Aplicar K-Means
kmeans.fit(df_std)

# Obtener las etiquetas de cluster asignadas a cada muestra
labels = kmeans.labels_

# Obtener las coordenadas de los centroides de los clusters
centroids = kmeans.cluster_centers_
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: Futur
eWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the v
alue of `n_init` explicitly to suppress the warning
```

In [ ]:
```
# Crear la gráfica de codo para determinar el número óptimo de clusters
WSSs = []
for i in range(1, 20):
    km = KMeans(n_clusters=i, random_state=0)
    km.fit(df_std)
    WSSs.append(km.inertia_)

plt.figure(figsize=(10, 6))
plt.plot(range(1, 20), WSSs, marker='o')
plt.title('Gráfica de Codo')
plt.xlabel('Número de Clusters')
plt.ylabel('Suma de las distancias cuadráticas dentro del cluster (WSS)')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: Fu
tureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set th
e value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: Fu
tureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set th
e value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: Fu
tureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set th
e value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: Fu
```

In [ ]:
```
from sklearn.metrics import silhouette_samples, silhouette_score

k = 2


kmeans = KMeans(n_clusters=k, random_state=0, n_init=10)
kmeans.fit(df_std)
y_clusters = kmeans.labels_
cluster_labels = np.unique(y_clusters)
silhouette_scores = silhouette_samples(df_std, y_clusters, metric='euclidean')

# Configurar la visualización del gráfico de silueta
y_ax_lower, y_ax_upper = 0, 0
yticks = []

# Iterar sobre los clusters
for i, c in enumerate(cluster_labels):
    silhouette_scores_c = silhouette_scores[y_clusters == c]
    silhouette_scores_c.sort()
    y_ax_upper += len(silhouette_scores_c)
    color = plt.cm.tab10(float(i) / k)  # Usar colores predeterminados
    plt.barh(range(y_ax_lower, y_ax_upper), silhouette_scores_c, height=1.0, e
    yticks.append((y_ax_lower + y_ax_upper) / 2.)
    y_ax_lower += len(silhouette_scores_c)

# Calcular y graficar la línea de silueta promedio
silhouette_avg = np.mean(silhouette_scores)
plt.axvline(silhouette_avg, color="black", linestyle="--")

plt.yticks(yticks, cluster_labels + 1)
plt.ylabel('Cluster')
plt.xlabel('Coeficiente de silueta')
plt.title('Gráfico de Silueta')
plt.tight_layout()
plt.show()
```



Se exploraron diferentes métricas y técnicas para determinar el número óptimo de clusters, sin embargo, debido a la falta de un codo significativo en el gráfico de codo y la presencia de valores negativos en el coeficiente de silueta, se presenta un desafío en la identificación clara de la estructura de clusters en los datos.

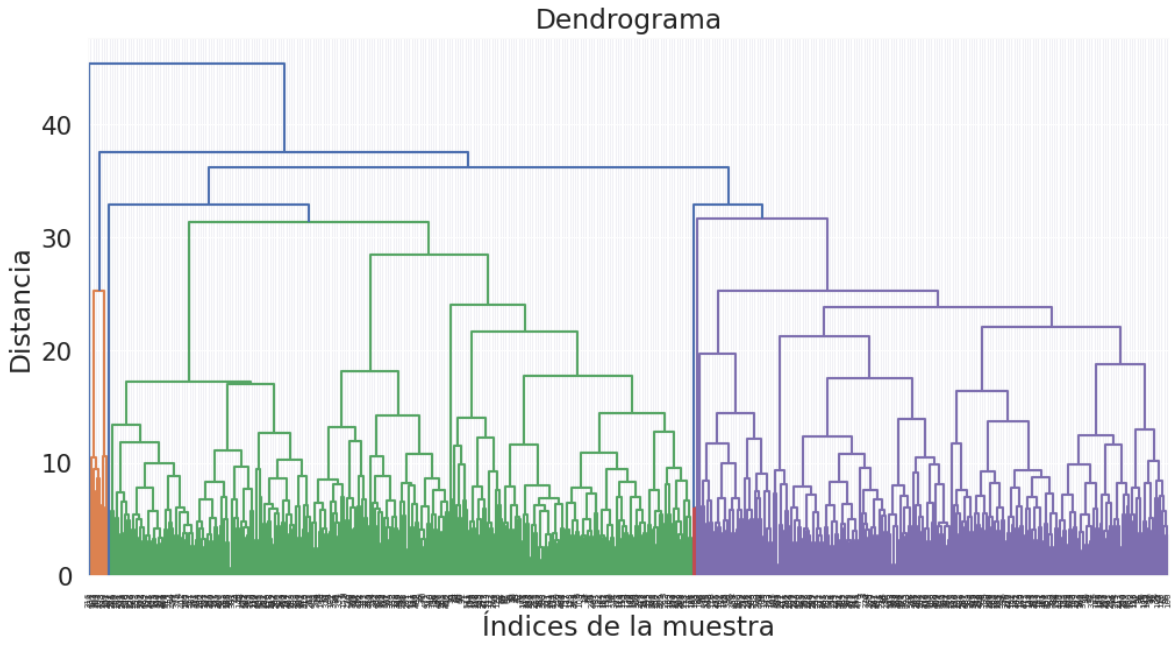
El gráfico de codo se utilizó para identificar el número óptimo de clusters en el algoritmo K-Means. Sin embargo, la representación visual de la suma de las distancias cuadradas intra-cluster (WSS) en función del número de clusters no mostró un punto de inflexión claro o "codo". Esta falta de un punto de codo sugiere que no hay una división clara de los datos en un número específico de clusters.

El coeficiente de silueta se utilizó como una métrica alternativa para evaluar la cohesión y la separación de los clusters. Se observó que algunas muestras tenían valores negativos en el coeficiente de silueta, lo que indica que podrían estar más cerca del centroide de un cluster vecino que del centroide de su propio cluster asignado. Esto pueder ser una posible superposición o ambigüedad en la asignación de clusters.

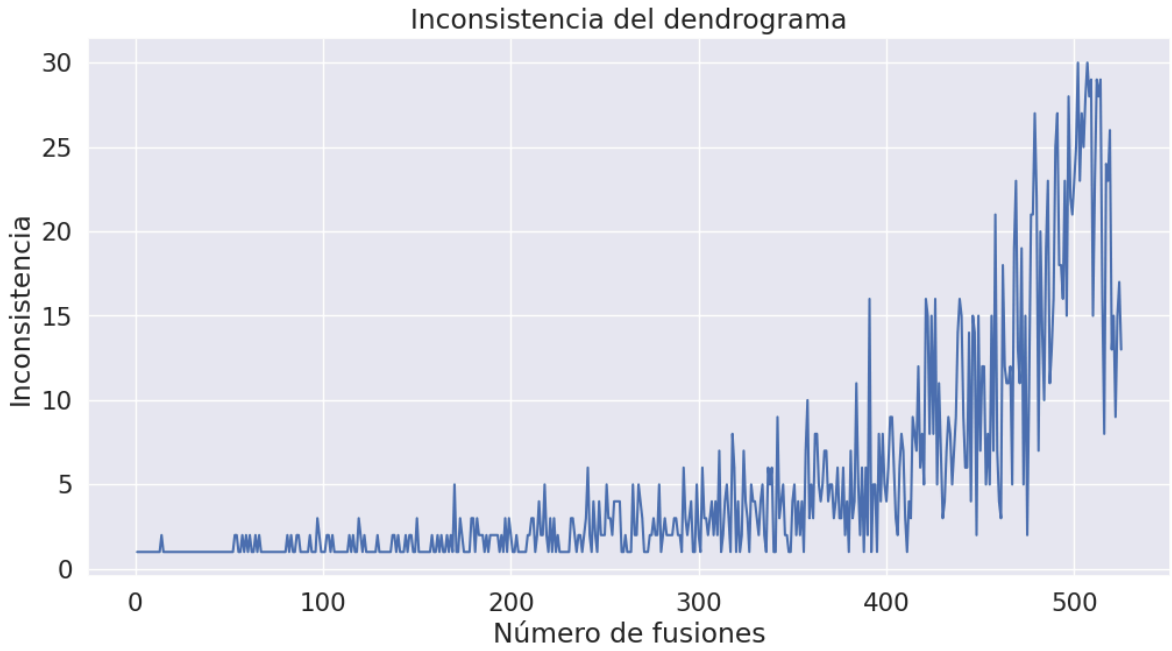
WARD (JERÁRQUICO)

In [ ]:
```
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import AgglomerativeClustering

Z = linkage(df_std, method='ward')
```

In [ ]:
```
# Dendrograma
plt.figure(figsize=(12, 6))
dendrogram(Z)
plt.title('Dendrograma')
plt.xlabel('Índices de la muestra')
plt.ylabel('Distancia')
plt.show()
```



In [ ]:
```
from scipy.cluster.hierarchy import dendrogram, linkage, inconsistent

# Calcular la inconsistencia
depth = 5  # Profundidad de cálculo de la inconsistencia
incons = inconsistent(Z, depth)

# Graficar la inconsistencia
plt.figure(figsize=(12, 6))
plt.plot(range(1, len(incons) + 1), incons[:, 2])
plt.title('Inconsistencia del dendrograma')
plt.xlabel('Número de fusiones')
plt.ylabel('Inconsistencia')
plt.show()
```



Se exploró la estructura de clustering mediante la construcción y análisis de un dendrograma. Basándonos en la observación del dendrograma, se sugiere que un número adecuado de clusters podría ser 5.

Análisis del Dendrograma: Al observar el dendrograma generado, se puede notar que hay un punto en el cual los clusters se dividen en cuatro grupos distintos. Este punto se caracteriza por una fusión significativa en la altura de las ramas, indicando una potencial división natural del dataset en 5 clusters.

Los puntos del dataset se agrupan en cuatro clusters distintos, cada uno representando una subpoblación potencialmente diferente en el contexto del Síndrome de Ovarios Poliquísticos.

7. Bibliografía

- American College of Obstetricians and Gynecologists. (2015). Polycystic ovary syndrome. Obtenido el 20 de mayo de 2016 en http://www.acog.org/Patients/FAQs/Polycystic-Ovary-Syndrome-PCOS (http://www.acog.org/Patients/FAQs/Polycystic-Ovary-Syndrome-PCOS) en el contenido de Inglés Notificación de salida

- U.S. Department of Health and Human Services, Office on Women's Health. (2014). Polycystic ovary syndrome (PCOS) fact sheet. Obtenido el 20 de mayo de 2016 en https://espanol.womenshealth.gov/a-z-topics/polycystic-ovary-syndrome (https://espanol.womenshealth.gov/a-z-topics/polycystic-ovary-syndrome)
- FONTES, R.; et al. Reference interval of thyroid stimulating hormone and free thyroxine in a reference population over 60 years old and in very old subjects (over 80 years): comparison to young subjects. Thyroid Res. 6. 13, 2013
- WARD, L. S. Devemos mudar os valores de referência para TSH normal?. Arq Bras Endocrinol Metab. 52. 1; 2008

Diseño asisitido por IA