Predicción del trastorno del espectro autista (TEA) en adultos usando ML

September 16, 2024

1 Estudio sobre el Trastorno del Espectro autista (TEA) en adultos

1.1 ÍNDICE

1.2 INTRODUCCIÓN

El **trastorno del Espectro Autista (TEA)** es una condición del neurodesarrollo que afecta la comunicación, la interacción social y el comportamiento. La detección temprana y precisa del TEA en adultos es crucial para brindarles el apoyo y las planificar la intervención necesaria agenciada a mejorar su calidad de vida.

Este proyecto tiene como objetivo investigar sobre las variables que intervienen y desarrollar modelos de aprendizaje automático capaces de predecir el riesgo de TEA en adultos a partir de datos demográficos y respuestas a un cuestionario de cribado.

Utilizaremos el conjunto de datos "Autism Screening Adult" del repositorio UCI Machine Learning, que contiene información de 704 adultos, incluyendo sus puntuaciones en el test AQ (Autism Spectrum Quotient) y un diagnóstico clínico de TEA.

Nos centraremos en la preparación y análisis exploratorio de los datos, seguido de una estrategia de modelado múltiple. Adaptaremos el conjunto de datos original optimizado para cada tipo específico de modelo de aprendizaje automático:

1. Modelo de Regresión Lineal-clasificación Binaria: Este modelo se centrará en predecir si un individuo tiene o no TEA, simplificando el problema a una clasificación de dos clases.
2. Modelo de Clasificación Multiclase: Iremos más allá de la clasificación binaria, intentando predecir diferentes niveles de riesgo de TEA, lo que podría ofrecer una evaluación más detallada y personalizada.

Para cada modelo, seleccionaremos y entrenaremos algoritmos específicos, evaluando su rendimiento y comparando sus resultados. Exploraremos técnicas como la regresión logística, los árboles de decisión y las máquinas de vectores de soporte, buscando identificar el modelo más preciso y adecuado para cada enfoque.

Como avanzabamos anteriormente además de desarrollar modelos predictivos, este proyecto busca contribuir a la comprensión de los ***factores que influyen en el riesgo de TEA en adultos***. Analizaremos la importancia de las diferentes características y cómo se relacionan con el diagnóstico de TEA, lo que podría aportar información valiosa para futuras investigaciones y estrategias de intervención.

Keywords:

#Trastorno del Espectro Autista, #aprendizaje automático, #clasificación binaria, #regresión lineal, #clasificación multiclase, #test AQ, #diagnóstico, #predicción*.

1.3 ETL (Extracción, Transformación y Carga).

1.3.1 EXTRACCIÓN

Extracción y Carga Inicial La fuente de datos para este proyecto es el conjunto de datos "Autism Screening Adult", disponible públicamente en el repositorio UCI Machine Learning. Los datos se proporcionaron en formato ARFF (Attribute-Relation File Format), un formato común en minería de datos que permite almacenar metadatos junto con los datos.

Para facilitar su manipulación y análisis en Python, se procedió a la carga de los datos desde el archivo ARFF a un DataFrame de Pandas, una estructura de datos tabular ampliamente utilizada

en ciencia de datos. Esta etapa inicial de extracción y carga nos permitió acceder a los datos y comenzar el proceso de transformación y exploración.

[ ]:
```
# importamos las librerias necesarias
import tensorflow as tf
import pandas as pd
import keras as kr
import scipy as scp
import numpy as np
```

Cargamos el conjunto de datos, originalmente en formato `.arff`:

[3]:
```
# Importamos el dataset en formato arff
from scipy.io import arff
dataSet, meta = arff.loadarff("autism_adult.arff")
# Observamos que tenemos un array de numpy
type(dataSet)
```

[3]:
```
numpy.ndarray
```

1.3.2 Preprocesamiento de Datos

Dado que el conjunto de datos puede contener valores faltantes o inconsistentes, es crucial realizar una limpieza y preprocesamiento antes de proceder con el análisis y modelado. Pandas, una biblioteca de análisis de datos en Python, ofrece herramientas eficientes para esta tarea.

Carga de Datos

Comenzamos cargando los datos desde el archivo CSV preprocesado en un DataFrame de Pandas, que nos proporciona una estructura tabular flexible para manipular y explorar los datos.

[4]:
```
# Convertimos en un dataframe de pandas:
df = pd.DataFrame(dataSet)
# Mostramos los primeros registros
df.head()
```

[4]:

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	\
0	b'1'	b'1'	b'1'	b'1'	b'0'	b'0'	b'1'	b'1'	
1	b'1'	b'1'	b'0'	b'1'	b'0'	b'0'	b'0'	b'1'	
2	b'1'	b'1'	b'0'	b'1'	b'1'	b'0'	b'1'	b'1'	
3	b'1'	b'1'	b'0'	b'1'	b'0'	b'0'	b'1'	b'1'	
4	b'1'	b'0'	b'0'	b'0'	b'0'	b'0'	b'0'	b'1'	

	A9_Score	A10_Score	…	gender	ethnicity	jundice	austim	\
0	b'0'	b'0'	…	b'f'	b'White-European'	b'no'	b'no'	
1	b'0'	b'1'	…	b'm'	b'Latino'	b'no'	b'yes'	
2	b'1'	b'1'	…	b'm'	b'Latino'	b'yes'	b'yes'	
3	b'0'	b'1'	…	b'f'	b'White-European'	b'no'	b'yes'	
4	b'0'	b'0'	…	b'f'	b'?'	b'no'	b'no'	

```
      contry_of_res used_app_before result        age_desc   relation  \
0  b'United States'           b'no'    6.0  b'18 and more'    b'Self'
1         b'Brazil'           b'no'    5.0  b'18 and more'    b'Self'
2          b'Spain'           b'no'    8.0  b'18 and more'  b'Parent'
3  b'United States'           b'no'    6.0  b'18 and more'    b'Self'
4          b'Egypt'           b'no'    2.0  b'18 and more'       b'?'

  Class/ASD
0     b'NO'
1     b'NO'
2    b'YES'
3     b'NO'
4     b'NO'

[5 rows x 21 columns]
```

1.3.3 TRANSFORMACIÓN

Transformación de los Datos En esta fase crucial, se llevaron a cabo una serie de operaciones y técnicas destinadas a limpiar, modificar y preparar los datos extraídos del archivo ARFF original. El propósito principal de esta etapa fue asegurar la calidad, consistencia y adecuación de los datos para su posterior análisis y utilización en modelos de aprendizaje automático.

Se abordaron problemas comunes en la calidad de los datos, como valores faltantes e inconsistencias en las categorías. Además, se implementaron estrategias de codificación para transformar las variables categóricas en un formato numérico apto para los algoritmos de aprendizaje automático. La ingeniería de características también jugó un papel importante en esta fase, permitiendo la creación de nuevas variables y la transformación de las existentes para potenciar su capacidad predictiva.

Finalmente, se aplicaron técnicas de remuestreo para abordar el desequilibrio de clases presente en la variable objetivo, asegurando una representación más equitativa de las diferentes categorías en el conjunto de datos.

[5]:
```
# Convertimos las columnas a string
# para poder solucionar el problema
# tipico del formato 'weka
for column in df.columns:
    if df[column].dtype == 'object':
        df[column] = df[column].astype(str)
# Comprobamos el formato de los registros
df.head()
```

[5]:
```
   A1_Score A2_Score A3_Score A4_Score A5_Score A6_Score A7_Score A8_Score  \
0         1         1         1         1         0         0         1         1
1         1         1         0         1         0         0         0         1
2         1         1         0         1         1         0         1         1
3         1         1         0         1         0         0         1         1
```

```
4         1         0         0         0         0         0         0         1

   A9_Score A10_Score  …  gender       ethnicity jundice austim  \
0         0          0  …       f  White-European      no     no
1         0          1  …       m          Latino      no    yes
2         1          1  …       m          Latino     yes    yes
3         0          1  …       f  White-European      no    yes
4         0          0  …       f               ?      no     no

    contry_of_res used_app_before result     age_desc relation Class/ASD
0   United States              no    6.0  18 and more     Self        NO
1          Brazil              no    5.0  18 and more     Self        NO
2           Spain              no    8.0  18 and more   Parent       YES
3   United States              no    6.0  18 and more     Self        NO
4           Egypt              no    2.0  18 and more        ?        NO

[5 rows x 21 columns]
```

[6]:
```
# Comprobamos si hay valores null en el DataFrame
# Porcentaje de valores nulos en el dataset
df.isnull().sum()/len(df)*100
```

[6]:
```
A1_Score           0.000000
A2_Score           0.000000
A3_Score           0.000000
A4_Score           0.000000
A5_Score           0.000000
A6_Score           0.000000
A7_Score           0.000000
A8_Score           0.000000
A9_Score           0.000000
A10_Score          0.000000
age                0.284091
gender             0.000000
ethnicity          0.000000
jundice            0.000000
austim             0.000000
contry_of_res      0.000000
used_app_before    0.000000
result             0.000000
age_desc           0.000000
relation           0.000000
Class/ASD          0.000000
dtype: float64
```

[7]:
```
# Visualizamos con otro metodo:
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 704 entries, 0 to 703
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   A1_Score         704 non-null    object
 1   A2_Score         704 non-null    object
 2   A3_Score         704 non-null    object
 3   A4_Score         704 non-null    object
 4   A5_Score         704 non-null    object
 5   A6_Score         704 non-null    object
 6   A7_Score         704 non-null    object
 7   A8_Score         704 non-null    object
 8   A9_Score         704 non-null    object
 9   A10_Score        704 non-null    object
 10  age              702 non-null    float64
 11  gender           704 non-null    object
 12  ethnicity        704 non-null    object
 13  jundice          704 non-null    object
 14  austim           704 non-null    object
 15  contry_of_res    704 non-null    object
 16  used_app_before  704 non-null    object
 17  result           704 non-null    float64
 18  age_desc         704 non-null    object
 19  relation         704 non-null    object
 20  Class/ASD        704 non-null    object
dtypes: float64(2), object(19)
memory usage: 115.6+ KB
```

Eliminamos valores ausentes en la variable edad ('age'). Al ser un porcentaje minimo (0.28%) de valores asusentes (2 registros), procedimos a eliminar las filas correspondientes:

[8]:
```
# eliminamos los registros con valores null:
df = df.dropna()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 702 entries, 0 to 703
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   A1_Score         702 non-null    object
 1   A2_Score         702 non-null    object
 2   A3_Score         702 non-null    object
 3   A4_Score         702 non-null    object
 4   A5_Score         702 non-null    object
 5   A6_Score         702 non-null    object
 6   A7_Score         702 non-null    object
```

```
 7   A8_Score          702 non-null    object
 8   A9_Score          702 non-null    object
 9   A10_Score         702 non-null    object
 10  age               702 non-null    float64
 11  gender            702 non-null    object
 12  ethnicity         702 non-null    object
 13  jundice           702 non-null    object
 14  austim            702 non-null    object
 15  contry_of_res     702 non-null    object
 16  used_app_before   702 non-null    object
 17  result            702 non-null    float64
 18  age_desc          702 non-null    object
 19  relation          702 non-null    object
 20  Class/ASD         702 non-null    object
dtypes: float64(2), object(19)
memory usage: 120.7+ KB
```

Observamos que se han eliminado dos registros.
Procedemos a limpiar los nombres de campos o atributos convirtiéndolos a minúsculas y normalizándolos:

[9]:
```
# Convertimos a minusculas los nombres de las columnas:
df.columns = df.columns.str.lower()
print(df.columns)
```

```
Index(['a1_score', 'a2_score', 'a3_score', 'a4_score', 'a5_score', 'a6_score',
       'a7_score', 'a8_score', 'a9_score', 'a10_score', 'age', 'gender',
       'ethnicity', 'jundice', 'austim', 'contry_of_res', 'used_app_before',
       'result', 'age_desc', 'relation', 'class/asd'],
      dtype='object')
```

[10]:
```
# Normalizamos los nombres de campo
df.columns = df.columns.str.replace('[^a-zA-Z0-9]', '_', regex=True)
print(df.columns)
```

```
Index(['a1_score', 'a2_score', 'a3_score', 'a4_score', 'a5_score', 'a6_score',
       'a7_score', 'a8_score', 'a9_score', 'a10_score', 'age', 'gender',
       'ethnicity', 'jundice', 'austim', 'contry_of_res', 'used_app_before',
       'result', 'age_desc', 'relation', 'class_asd'],
      dtype='object')
```

El Cuestionario AQ-10 y su Representación en el Conjunto de Datos
El Autism-Spectrum Quotient (AQ-10) es un cuestionario de autoinforme compuesto por 10 preguntas, diseñado para evaluar la presencia de rasgos autistas en adultos. Cada pregunta ofrece opciones de respuesta que van desde "Definitivamente de acuerdo" hasta "Definitivamente en desacuerdo", y cada respuesta se codifica con un valor numérico. La suma de estas puntuaciones individuales proporciona una puntuación total, la cual se interpreta como un indicador de la probabilidad de que un individuo presente rasgos autistas.

En el conjunto de datos utilizado los atributos "A1_Score", "A2_Score", etc., representan las

puntuaciones obtenidas en cada una de las 10 preguntas del AQ-10. Dado que estas puntuaciones son inherentemente numéricas, se realizó una conversión de aquellas columnas que, por algún motivo en la carga de datos, se encontraban en formato 'object' o 'float' a un tipo numérico entero (int) para asegurar su correcta utilización en los análisis posteriores.

[11]:
```
# importamos la libreria para expresiones regulares
import re

# Convertimos a int
for column in df.columns:
    if re.match(r'^a\d+', column) and df[column].dtype == 'object':
        try:
            df[column] = pd.to_numeric(df[column])
        except ValueError:
            print(f"Advertencia: La columna'{column}'contiene valores"
                  "no numéricos y no se pudo convertir")
# Revisamos que los campos han cambiado de tipo adecuadamente
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 702 entries, 0 to 703
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   a1_score         702 non-null    int64
 1   a2_score         702 non-null    int64
 2   a3_score         702 non-null    int64
 3   a4_score         702 non-null    int64
 4   a5_score         702 non-null    int64
 5   a6_score         702 non-null    int64
 6   a7_score         702 non-null    int64
 7   a8_score         702 non-null    int64
 8   a9_score         702 non-null    int64
 9   a10_score        702 non-null    int64
 10  age              702 non-null    float64
 11  gender           702 non-null    object
 12  ethnicity        702 non-null    object
 13  jundice          702 non-null    object
 14  austim           702 non-null    object
 15  contry_of_res    702 non-null    object
 16  used_app_before  702 non-null    object
 17  result           702 non-null    float64
 18  age_desc         702 non-null    object
 19  relation         702 non-null    object
 20  class_asd        702 non-null    object
dtypes: float64(2), int64(10), object(9)
memory usage: 120.7+ KB
```

Procedimos a verificar la presencia de valores atípicos o extremos en la columna age.

[12]:
```
# Marcamos valores mayores a 115 como NaN (Not a Number)
df.loc[df['age'] > 115, 'age'] = float('nan')

# Eliminamos filas con valores NaN en la columna 'age'
df.dropna(subset=['age'], inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 701 entries, 0 to 703
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   a1_score         701 non-null    int64
 1   a2_score         701 non-null    int64
 2   a3_score         701 non-null    int64
 3   a4_score         701 non-null    int64
 4   a5_score         701 non-null    int64
 5   a6_score         701 non-null    int64
 6   a7_score         701 non-null    int64
 7   a8_score         701 non-null    int64
 8   a9_score         701 non-null    int64
 9   a10_score        701 non-null    int64
 10  age              701 non-null    float64
 11  gender           701 non-null    object
 12  ethnicity        701 non-null    object
 13  jundice          701 non-null    object
 14  austim           701 non-null    object
 15  contry_of_res    701 non-null    object
 16  used_app_before  701 non-null    object
 17  result           701 non-null    float64
 18  age_desc         701 non-null    object
 19  relation         701 non-null    object
 20  class_asd        701 non-null    object
dtypes: float64(2), int64(10), object(9)
memory usage: 120.5+ KB
```

Se eliminó un valor atípico identificado en el atributo age. Posteriormente, se convirtió dicho atributo a tipo numérico entero.

[13]:
```
# Cambiamos el campo edad a 'int':
df['age'] = df['age'].astype(int)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 701 entries, 0 to 703
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype
```

```
---  ------           --------------  -----
 0   a1_score         701 non-null    int64
 1   a2_score         701 non-null    int64
 2   a3_score         701 non-null    int64
 3   a4_score         701 non-null    int64
 4   a5_score         701 non-null    int64
 5   a6_score         701 non-null    int64
 6   a7_score         701 non-null    int64
 7   a8_score         701 non-null    int64
 8   a9_score         701 non-null    int64
 9   a10_score        701 non-null    int64
 10  age              701 non-null    int64
 11  gender           701 non-null    object
 12  ethnicity        701 non-null    object
 13  jundice          701 non-null    object
 14  austim           701 non-null    object
 15  contry_of_res    701 non-null    object
 16  used_app_before  701 non-null    object
 17  result           701 non-null    float64
 18  age_desc         701 non-null    object
 19  relation         701 non-null    object
 20  class_asd        701 non-null    object
dtypes: float64(1), int64(11), object(9)
memory usage: 120.5+ KB
```

Procedimos a revisar los atributos una vez más:

[14]:
```
# Obtenemos las columnas
print(df.columns)
```

```
Index(['a1_score', 'a2_score', 'a3_score', 'a4_score', 'a5_score', 'a6_score',
       'a7_score', 'a8_score', 'a9_score', 'a10_score', 'age', 'gender',
       'ethnicity', 'jundice', 'austim', 'contry_of_res', 'used_app_before',
       'result', 'age_desc', 'relation', 'class_asd'],
      dtype='object')
```

A continuación, se procedimos al análisis del campo gender, el cual presentaba dos categorías: “male” (masculino) y “female” (femenino).

[15]:
```
# Importamos la libreria sklern
from sklearn.preprocessing import LabelEncoder

# Creamos una variable
le = LabelEncoder()

# Etiquetamos la variable categorica 'gender':
df['gender'] = le.fit_transform(df['gender'])

# Imprimimos las clases:
```

```
print(le.classes_)
```

```
['f' 'm']
```

La columna gender se codificará de la siguiente manera: **0** para “Female” (femenino) y **1** para “Male” (masculino).

Proseguimos con el análisis de la variable categórica ethnicity. Para ello, comenzarmos examinando la distribución de sus valores en el DataFrame.

[16]:
```
# Dilucidamos el numero de etnias en el df
df['ethnicity'].value_counts()
```

[16]:
```
ethnicity
White-European     233
Asian              123
?                   93
Middle Eastern      92
Black               43
South Asian         36
Others              30
Latino              20
Hispanic            13
Pasifika            11
Turkish              6
others               1
Name: count, dtype: int64
```

Se observó la presencia del carácter ‘?’ en la columna ethnicity, el cual denota datos faltantes o desconocidos. Se decidió agrupar estos valores en la categoría ‘Others’ para simplificar el análisis. Además, se identificó una inconsistencia en la categoría ‘others’, la cual se unificó con ‘Others’ para mantener la coherencia en los datos.

A continuación, se procedió a realizar transformaciones adicionales en las categorías de la variable ethnicity.

[17]:
```
# Reemplazamos valores específicos en la columna 'ethnicity'
df['ethnicity'] = df['ethnicity'].replace(['?', 'others'], 'Others')
# Y observamos si hemos agrupado todo bajo la categoria 'Others'
df['ethnicity'].value_counts()
```

[17]:
```
ethnicity
White-European     233
Others             124
Asian              123
Middle Eastern      92
Black               43
South Asian         36
Latino              20
```

```
Hispanic            13
Pasifika            11
Turkish              6
Name: count, dtype: int64
```

Algunas categorías parecían redundantes, como es el caso de 'Hispanic' y 'Latino', por lo que procedimos a realizar agrupaciones:

[18]:
```
# Reemplazamos otros valores en la columna 'ethnicity'
df['ethnicity'] = df['ethnicity'].replace(['Hispanic', 'Latino'], 'Latin')
df['ethnicity'] = df['ethnicity'].replace('White-European', 'White')
# Y observamos si hemos agrupado todo bajo la categorias antes especificadas
df['ethnicity'].value_counts()
```

[18]:
```
ethnicity
White              233
Others             124
Asian              123
Middle Eastern      92
Black               43
South Asian         36
Latin               33
Pasifika            11
Turkish              6
Name: count, dtype: int64
```

Incluimos las categorías con menos registros 'Pasifika' y 'Turkish' en 'Others' para evitar problemas con el modelo:

[19]:
```
# Agrupamos registros de etnias minoritarias
df['ethnicity'] = df['ethnicity'].replace(['Pasifika', 'Turkish'], 'Others')
# Observamos si hemos agrupado
df['ethnicity'].value_counts()
```

[19]:
```
ethnicity
White              233
Others             141
Asian              123
Middle Eastern      92
Black               43
South Asian         36
Latin               33
Name: count, dtype: int64
```

[20]:
```
# Limpiamos la columna 'ethnicity' de espacios en blanco
df['ethnicity'] = df['ethnicity'].astype(str).str.strip()
```

Mapeamos las categorías dentro del atributo 'etnicidad' creando un diccionario:

[21]:
```
# mapeando los valores
eth_map = {
    "White": 1,
    "Asian": 2,
    "Middle Eastern": 3,
    "Black": 4,
    "South Asian": 5,
    "Latin": 6,
    "Others": 7
}
# comprobamos los pares clave - valor
print(eth_map)
```

```
{'White': 1, 'Asian': 2, 'Middle Eastern': 3, 'Black': 4, 'South Asian': 5,
'Latin': 6, 'Others': 7}
```

[22]:
```
# Asignamos el resultado del mapeo a la columna 'eth_cat'
df['eth_cat'] = df['ethnicity'].map(eth_map)
df[['ethnicity', 'eth_cat']].head(10)
```

[22]:

	ethnicity	eth_cat
0	White	1
1	Latin	6
2	Latin	6
3	White	1
4	Others	7
5	Others	7
6	Black	4
7	White	1
8	White	1
9	Asian	2

[23]:
```
# Aseguramos que la columna 'eth_cat' sea de tipo int64
df['eth_cat'] = df['eth_cat'].astype('int64')
# Observamos los resultados:
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 701 entries, 0 to 703
Data columns (total 22 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   a1_score        701 non-null    int64
 1   a2_score        701 non-null    int64
 2   a3_score        701 non-null    int64
 3   a4_score        701 non-null    int64
 4   a5_score        701 non-null    int64
 5   a6_score        701 non-null    int64
```

```
 6   a7_score          701 non-null    int64
 7   a8_score          701 non-null    int64
 8   a9_score          701 non-null    int64
 9   a10_score         701 non-null    int64
 10  age               701 non-null    int64
 11  gender            701 non-null    int64
 12  ethnicity         701 non-null    object
 13  jundice           701 non-null    object
 14  austim            701 non-null    object
 15  contry_of_res     701 non-null    object
 16  used_app_before   701 non-null    object
 17  result            701 non-null    float64
 18  age_desc          701 non-null    object
 19  relation          701 non-null    object
 20  class_asd         701 non-null    object
 21  eth_cat           701 non-null    int64
dtypes: float64(1), int64(13), object(8)
memory usage: 126.0+ KB
```

De cara a la creación de modelos de ML, la variable categórica 'ethnicity' parecía no tener cabida utilidad, por lo que procedemos a eliminarla:

[24]:
```
# Eliminamos la columna 'ethnicity'
df.drop(columns=['ethnicity'], inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 701 entries, 0 to 703
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   a1_score          701 non-null    int64
 1   a2_score          701 non-null    int64
 2   a3_score          701 non-null    int64
 3   a4_score          701 non-null    int64
 4   a5_score          701 non-null    int64
 5   a6_score          701 non-null    int64
 6   a7_score          701 non-null    int64
 7   a8_score          701 non-null    int64
 8   a9_score          701 non-null    int64
 9   a10_score         701 non-null    int64
 10  age               701 non-null    int64
 11  gender            701 non-null    int64
 12  jundice           701 non-null    object
 13  austim            701 non-null    object
 14  contry_of_res     701 non-null    object
 15  used_app_before   701 non-null    object
 16  result            701 non-null    float64
```

```
 17  age_desc         701 non-null    object
 18  relation         701 non-null    object
 19  class_asd        701 non-null    object
 20  eth_cat          701 non-null    int64
dtypes: float64(1), int64(13), object(7)
memory usage: 120.5+ KB
```

Se consideró necesario convertir los resultados del test AQ-10 a un formato binario para facilitar la identificación de posibles casos de autismo. Según los criterios del test, una puntuación igual o superior a 6 indica la necesidad de una evaluación especializada, mientras que una puntuación inferior a 6 sugiere una baja probabilidad de TEA.

En consecuencia, se creó una nueva variable binaria denominada aq_binary, donde los resultados iguales o mayores a 6 se codificaron como True (1), y los resultados inferiores a 6 como False (0). Esta transformación permitirá utilizar esta variable como un indicador simplificado del riesgo de autismo en análisis posteriores.

[25]:
```
# Creamos una variable binaria para AQ
df['aq_binary'] = (df['result'] >= 6).astype('int64')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 701 entries, 0 to 703
Data columns (total 22 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   a1_score         701 non-null    int64
 1   a2_score         701 non-null    int64
 2   a3_score         701 non-null    int64
 3   a4_score         701 non-null    int64
 4   a5_score         701 non-null    int64
 5   a6_score         701 non-null    int64
 6   a7_score         701 non-null    int64
 7   a8_score         701 non-null    int64
 8   a9_score         701 non-null    int64
 9   a10_score        701 non-null    int64
 10  age              701 non-null    int64
 11  gender           701 non-null    int64
 12  jundice          701 non-null    object
 13  austim           701 non-null    object
 14  contry_of_res    701 non-null    object
 15  used_app_before  701 non-null    object
 16  result           701 non-null    float64
 17  age_desc         701 non-null    object
 18  relation         701 non-null    object
 19  class_asd        701 non-null    object
 20  eth_cat          701 non-null    int64
 21  aq_binary        701 non-null    int64
dtypes: float64(1), int64(14), object(7)
```

```
memory usage: 126.0+ KB
```

Se exploró el atributo 'jundice', el cual registraba si el sujeto había nacido con ictericia. Posteriormente se llevó a cabo un análisis para determinar si existía alguna asociación entre la presencia de ictericia al nacer y el diagnóstico de autismo en la muestra.

[26]:
```
# Creamos una variable para las etiquetas ictericia
le = LabelEncoder()
# Etiquetamos la variable categorica 'jundice':
df['jundice'] = le.fit_transform(df['jundice'])
# Imprimimos las clases
print(le.classes_)
```

```
['no' 'yes']
```

Observamos que, efectivamente **'no'** ha quedado codificado como 0 y **'yes'** como 1, respetivamente. Comprobamos que efectivamente se han producido los cambios:

[27]:
```
# Comprobamos el atributo 'jundice':
df['jundice'].head(8)
```

[27]:
```
0    0
1    0
2    1
3    0
4    0
5    1
6    0
7    0
Name: jundice, dtype: int64
```

Se procedió a codificar la variable objetivo austim, la cual representa el diagnóstico de autismo.

[28]:
```
# re-creamos la variable
le = LabelEncoder()
# Etiquetamos la variable categorica 'austim':
df['austim'] = le.fit_transform(df['austim'])
# Imprimimos las clases
print(le.classes_)
```

```
['no' 'yes']
```

Se estableció la codificación de la variable austim, asignando el valor 0 a la categoría "no" y el valor 1 a la categoría "yes".

Posteriormente, se analizó la distribución de la variable objetivo austim para evaluar el equilibrio entre las clases (presencia o ausencia de diagnóstico de autismo) y verificar que el tipo de atributo fuera el adecuado para su uso en los modelos de aprendizaje automático.

[29]:
```
# Contamos los valores:
df['austim'].value_counts()
```

[29]:
```
austim
0    610
1     91
Name: count, dtype: int64
```

Aunque la transformación de los datos arrojó resultados prometedores, se identificó un desequilibrio de clases en la muestra, lo cual requirió la implementación de técnicas de remuestreo durante el entrenamiento de los modelos. Se optó por utilizar SMOTE (Técnica de Sobremuestreo de Minorías Sintéticas) o ADASYN (Muestreo Sintético Adaptativo), ambas capaces de generar muestras sintéticas de la clase minoritaria para equilibrar el conjunto de datos.

El siguiente atributo analizado fue country_of_res, referente al país de residencia del participante. Aunque inicialmente se consideró la posibilidad de que este atributo pudiera ser redundante dada la inclusión de la etnia en el análisis, se decidió conservarlo temporalmente para explorar su relación tanto con la etnia como con el diagnóstico de autismo durante la fase de análisis exploratorio de datos.

Para examinar si existía alguna asociación significativa entre estas variables, se emplearon técnicas como tablas de contingencia, pruebas de chi-cuadrado y visualizaciones.

A continuación, se presenta un análisis detallado del contenido de este campo.

[30]:
```
# Comprobamos el atributo 'contry_of_res':
df['contry_of_res'].value_counts()
```

[30]:
```
contry_of_res
United States           113
United Arab Emirates     82
India                    81
New Zealand              80
United Kingdom           77
                       ...
China                     1
Chile                     1
Lebanon                   1
Burundi                   1
Cyprus                    1
Name: count, Length: 67, dtype: int64
```

Dada la presencia de 67 países diferentes en el conjunto de datos, algunos con un número muy reducido de muestras, se evaluaron técnicas de codificación como `One-Hot Encoding` y `Mean Encoding`.

Sin embargo, considerando la necesidad de obtener un conjunto de datos compacto y adecuado para la aplicación de diversos modelos de aprendizaje automático, se optó por codificar cada país mediante `Label Encoding`.

[31]:
```
le = LabelEncoder()
df['res_code'] = le.fit_transform(df['contry_of_res'])

# Obtenemos el mapeo de etiquetas a códigos
code_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
print(code_mapping)
```

```
{'Afghanistan': 0, 'AmericanSamoa': 1, 'Angola': 2, 'Argentina': 3, 'Armenia':
4, 'Aruba': 5, 'Australia': 6, 'Austria': 7, 'Azerbaijan': 8, 'Bahamas': 9,
'Bangladesh': 10, 'Belgium': 11, 'Bolivia': 12, 'Brazil': 13, 'Burundi': 14,
'Canada': 15, 'Chile': 16, 'China': 17, 'Costa Rica': 18, 'Cyprus': 19, 'Czech
Republic': 20, 'Ecuador': 21, 'Egypt': 22, 'Ethiopia': 23, 'Finland': 24,
'France': 25, 'Germany': 26, 'Hong Kong': 27, 'Iceland': 28, 'India': 29,
'Indonesia': 30, 'Iran': 31, 'Iraq': 32, 'Ireland': 33, 'Italy': 34, 'Japan':
35, 'Jordan': 36, 'Kazakhstan': 37, 'Lebanon': 38, 'Malaysia': 39, 'Mexico': 40,
'Nepal': 41, 'Netherlands': 42, 'New Zealand': 43, 'Nicaragua': 44, 'Niger': 45,
'Oman': 46, 'Pakistan': 47, 'Philippines': 48, 'Portugal': 49, 'Romania': 50,
'Russia': 51, 'Saudi Arabia': 52, 'Serbia': 53, 'Sierra Leone': 54, 'South
Africa': 55, 'Spain': 56, 'Sri Lanka': 57, 'Sweden': 58, 'Tonga': 59, 'Turkey':
60, 'Ukraine': 61, 'United Arab Emirates': 62, 'United Kingdom': 63, 'United
States': 64, 'Uruguay': 65, 'Viet Nam': 66}
```

Exportamos los datos para posteriores consultas a formato JSON:

[32]:
```
# import la libreria json
import json
# Convertimos los valores a enteros de Python
for key, value in code_mapping.items():
    code_mapping[key] = int(value)

with open('country_code_mapping.json', 'w') as f:
    json.dump(code_mapping, f)
```

[33]:
```
# Revisamos la asignación de códigos
df[['contry_of_res', 'res_code']].head(10)
```

[33]:

	contry_of_res	res_code
0	United States	64
1	Brazil	13
2	Spain	56
3	United States	64
4	Egypt	22
5	United States	64
6	United States	64
7	New Zealand	43
8	United States	64
9	Bahamas	9

Una vez codificados los países, podemos eliminar el atributo 'country_of_res'

[34]:
```
# eliminamos 'country_of_res'
df.drop(columns=['contry_of_res'], inplace=True)
# Observamos los resultados:
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 701 entries, 0 to 703
Data columns (total 22 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   a1_score         701 non-null    int64
 1   a2_score         701 non-null    int64
 2   a3_score         701 non-null    int64
 3   a4_score         701 non-null    int64
 4   a5_score         701 non-null    int64
 5   a6_score         701 non-null    int64
 6   a7_score         701 non-null    int64
 7   a8_score         701 non-null    int64
 8   a9_score         701 non-null    int64
 9   a10_score        701 non-null    int64
 10  age              701 non-null    int64
 11  gender           701 non-null    int64
 12  jundice          701 non-null    int64
 13  austim           701 non-null    int64
 14  used_app_before  701 non-null    object
 15  result           701 non-null    float64
 16  age_desc         701 non-null    object
 17  relation         701 non-null    object
 18  class_asd        701 non-null    object
 19  eth_cat          701 non-null    int64
 20  aq_binary        701 non-null    int64
 21  res_code         701 non-null    int64
dtypes: float64(1), int64(17), object(4)
memory usage: 126.0+ KB
```

El siguiente atributo considerado para el análisis fue used_app_before, el cual indicaba si el participante había utilizado previamente alguna aplicación relacionada con el autismo. Se evaluó la relevancia de este atributo en el contexto del estudio, teniendo en cuenta las siguientes consideraciones:

- Familiaridad con el tema: El uso previo de una aplicación podría sugerir un mayor conocimiento sobre el autismo o una búsqueda activa de información o ayuda relacionada.
- Posible sesgo de selección: La utilización de aplicaciones podría indicar diferencias entre los participantes que las usaron y los que no, lo que podría introducir sesgos en los resultados.
- Potencial efecto de la aplicación: Si la aplicación utilizada fue una herramienta de intervención, su uso previo podría estar asociado a cambios en el comportamiento o los síntomas del participante.

[35]:
```
# Contamos los valores:
df['used_app_before'].value_counts()
```

[35]:
```
used_app_before
no     689
yes     12
Name: count, dtype: int64
```

Dada la pequeña cantidad de casos "yes" y el potencial de sesgo, lo recomendable es eliminar los casos "yes" en esta etapa inicial del análisis y la variable. Esto permitirá construir modelos más robustos y generalizables basados en un grupo más homogéneo de sujetos.

[36]:
```
# Filtramos los registros que no han usado la app:
df = df[df['used_app_before'] == 'no'].copy()

# Eliminamos la columna 'used_app_before'
df.drop(columns=['used_app_before'], inplace=True)

# Observamos los resultados:
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 689 entries, 0 to 703
Data columns (total 21 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   a1_score    689 non-null    int64
 1   a2_score    689 non-null    int64
 2   a3_score    689 non-null    int64
 3   a4_score    689 non-null    int64
 4   a5_score    689 non-null    int64
 5   a6_score    689 non-null    int64
 6   a7_score    689 non-null    int64
 7   a8_score    689 non-null    int64
 8   a9_score    689 non-null    int64
 9   a10_score   689 non-null    int64
 10  age         689 non-null    int64
 11  gender      689 non-null    int64
 12  jundice     689 non-null    int64
 13  austim      689 non-null    int64
 14  result      689 non-null    float64
 15  age_desc    689 non-null    object
 16  relation    689 non-null    object
 17  class_asd   689 non-null    object
 18  eth_cat     689 non-null    int64
 19  aq_binary   689 non-null    int64
 20  res_code    689 non-null    int64
dtypes: float64(1), int64(17), object(3)
```

```
memory usage: 118.4+ KB
```

Convertimos el atributo `result` un numero entero:

[37]:
```
# convertirmos 'result' como entero
df['result'] = df['result'].astype("int64")
# Revisamos los resultados
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 689 entries, 0 to 703
Data columns (total 21 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   a1_score   689 non-null    int64
 1   a2_score   689 non-null    int64
 2   a3_score   689 non-null    int64
 3   a4_score   689 non-null    int64
 4   a5_score   689 non-null    int64
 5   a6_score   689 non-null    int64
 6   a7_score   689 non-null    int64
 7   a8_score   689 non-null    int64
 8   a9_score   689 non-null    int64
 9   a10_score  689 non-null    int64
 10  age        689 non-null    int64
 11  gender     689 non-null    int64
 12  jundice    689 non-null    int64
 13  austim     689 non-null    int64
 14  result     689 non-null    int64
 15  age_desc   689 non-null    object
 16  relation   689 non-null    object
 17  class_asd  689 non-null    object
 18  eth_cat    689 non-null    int64
 19  aq_binary  689 non-null    int64
 20  res_code   689 non-null    int64
dtypes: int64(18), object(3)
memory usage: 118.4+ KB
```

A continuación exploraramos la variable 'relation' que se refería a quién proporcionó la información para el registro:

[38]:
```
# Exploramos los registros:
df['relation'].value_counts()
```

[38]:
```
relation
Self                        513
?                            91
Parent                       49
Relative                     27
```

```
Others                          5
Health care professional        4
Name: count, dtype: int64
```

Vamos a agrupar las diferentes relaciones para un mejor funcionamiento de los modelos: - valores '?' a 'Health care professional' - agrupación de categorías pequeñas

[39]:
```
# Vamos a asignar los valores '?' a 'Health care professional':
df['relation'] = df['relation'].replace('?', 'Health care professional')
# Agrupamos categorías pequeñas
df['relation'] = df['relation'].replace(['Others', 'Health care professional'],
                                        'Relative')
# Volvemos a contar valores
df['relation'].value_counts()
```

[39]:
```
relation
Self        513
Relative    127
Parent       49
Name: count, dtype: int64
```

Tras la agrupación de categorías, se constató una distribución más equilibrada en la variable relation, reflejando una mayor participación de familiares y una menor presencia de padres en el proceso de recopilación de datos. Este hallazgo se alinea con la naturaleza del conjunto de datos, centrado en adultos diagnosticados con autismo, donde la participación de los padres tiende a ser menor en comparación con estudios en poblaciones más jóvenes.

Posteriormente, se procedió a codificar el atributo relation para su inclusión en los modelos de aprendizaje automático.

[40]:
```
le = LabelEncoder()
df['relation_coded'] = le.fit_transform(df['relation'])

# Obtenemos el mapeo de etiquetas a códigos
code_mapping = dict(zip(le.classes_, le.transform(le.classes_)))

# Imprimimos el mapeo
print(code_mapping)

# Convertimos los valores a enteros de Python
for key, value in code_mapping.items():
    code_mapping[key] = int(value)

with open('relation_code_mapping.json', 'w') as f:
    json.dump(code_mapping, f)
```

```
{'Parent': 0, 'Relative': 1, 'Self': 2}
```

[41]:
```
# Eliminamos la columna 'relation'
df.drop(columns=['relation'], inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 689 entries, 0 to 703
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   a1_score        689 non-null    int64
 1   a2_score        689 non-null    int64
 2   a3_score        689 non-null    int64
 3   a4_score        689 non-null    int64
 4   a5_score        689 non-null    int64
 5   a6_score        689 non-null    int64
 6   a7_score        689 non-null    int64
 7   a8_score        689 non-null    int64
 8   a9_score        689 non-null    int64
 9   a10_score       689 non-null    int64
 10  age             689 non-null    int64
 11  gender          689 non-null    int64
 12  jundice         689 non-null    int64
 13  austim          689 non-null    int64
 14  result          689 non-null    int64
 15  age_desc        689 non-null    object
 16  class_asd       689 non-null    object
 17  eth_cat         689 non-null    int64
 18  aq_binary       689 non-null    int64
 19  res_code        689 non-null    int64
 20  relation_coded  689 non-null    int64
dtypes: int64(19), object(2)
memory usage: 118.4+ KB
```

Finalmente, se descartó el atributo age_desc, dado que la información sobre la edad de los participantes ya estaba disponible en la variable numérica age.

En relación a `class_asc`, al no encontrarse información en la documentación del repositorio UCI y contener únicamente valores binarios (“yes”/“no”), se tomó la decisión de eliminarlo del conjunto de datos.

[42]:
```
# Eliminamos las columnas
df.drop(columns=['age_desc', 'class_asd'], inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 689 entries, 0 to 703
Data columns (total 19 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
```

```
 0   a1_score         689 non-null    int64
 1   a2_score         689 non-null    int64
 2   a3_score         689 non-null    int64
 3   a4_score         689 non-null    int64
 4   a5_score         689 non-null    int64
 5   a6_score         689 non-null    int64
 6   a7_score         689 non-null    int64
 7   a8_score         689 non-null    int64
 8   a9_score         689 non-null    int64
 9   a10_score        689 non-null    int64
 10  age              689 non-null    int64
 11  gender           689 non-null    int64
 12  jundice          689 non-null    int64
 13  austim           689 non-null    int64
 14  result           689 non-null    int64
 15  eth_cat          689 non-null    int64
 16  aq_binary        689 non-null    int64
 17  res_code         689 non-null    int64
 18  relation_coded   689 non-null    int64
dtypes: int64(19)
memory usage: 107.7 KB
```

Creamos una copia de seguridad:

[51]:
```
# copia de seguridad del dataset en formato .csv
df.to_csv("autism_adult_cleaned.csv", index=False)
```

Análisis Exploratorio de Datos (EDA) y Garantía de Calidad de Datos (DQA) - Fase I Siguiendo la convención establecida, el Análisis Exploratorio de Datos (EDA) y la Garantía de Calidad de Datos (DQA) se llevaron a cabo como parte integral de la fase de Transformación dentro del proceso ETL. No obstante, cabe destacar que estas actividades también pueden realizarse de manera independiente, posterior a la etapa inicial de ETL.

Para llevar a cabo el EDA y el DQA, se emplearon diversas funciones integradas de la biblioteca `Pandas`, con el objetivo de obtener estadísticas descriptivas que permitieran una comprensión inicial de los datos.

[43]:
```
# 1. Utilizamos algunas funciones integradas de pandas
# para obtener estadísticas descriptivas del dataset
df.describe()
```

[43]:

	a1_score	a2_score	a3_score	a4_score	a5_score	a6_score	\
count	689.000000	689.000000	689.000000	689.000000	689.000000	689.000000	
mean	0.725689	0.454282	0.455733	0.496372	0.499274	0.280116	
std	0.446490	0.498267	0.498398	0.500350	0.500363	0.449382	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	

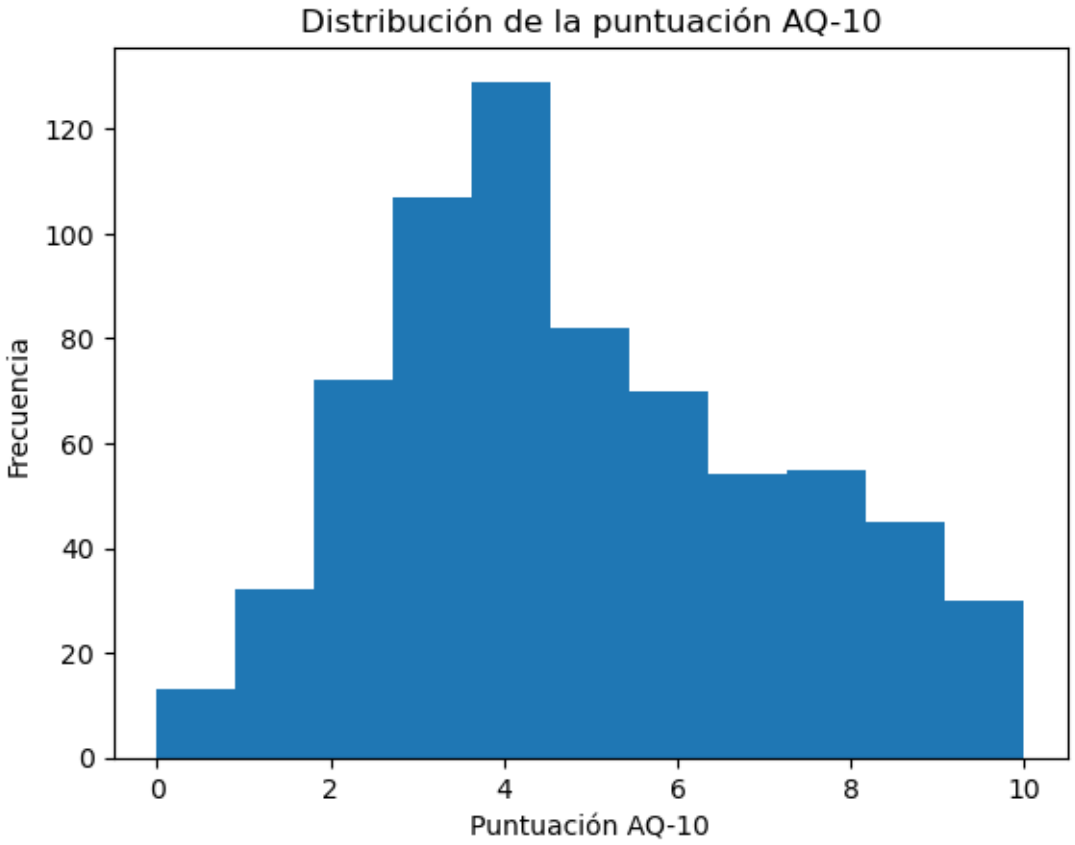
```
75%      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000
max      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000

         a7_score    a8_score    a9_score   a10_score         age      gender  \
count  689.000000  689.000000  689.000000  689.000000  689.000000  689.000000
mean     0.419448    0.654572    0.322206    0.577649   29.239478    0.523948
std      0.493827    0.475853    0.467661    0.494293    9.745116    0.499789
min      0.000000    0.000000    0.000000    0.000000   17.000000    0.000000
25%      0.000000    0.000000    0.000000    0.000000   21.000000    0.000000
50%      0.000000    1.000000    0.000000    1.000000   27.000000    1.000000
75%      1.000000    1.000000    1.000000    1.000000   35.000000    1.000000
max      1.000000    1.000000    1.000000    1.000000   64.000000    1.000000

          jundice      austim      result     eth_cat   aq_binary    res_code  \
count  689.000000  689.000000  689.000000  689.000000  689.000000  689.000000
mean     0.097242    0.129173    4.885341    3.253991    0.368650   44.095791
std      0.296503    0.335635    2.495328    2.311043    0.482789   19.263008
min      0.000000    0.000000    0.000000    1.000000    0.000000    0.000000
25%      0.000000    0.000000    3.000000    1.000000    0.000000   29.000000
50%      0.000000    0.000000    4.000000    2.000000    0.000000   43.000000
75%      0.000000    0.000000    7.000000    5.000000    1.000000   63.000000
max      1.000000    1.000000   10.000000    7.000000    1.000000   66.000000

       relation_coded
count      689.000000
mean         1.673440
std          0.602229
min          0.000000
25%          1.000000
50%          2.000000
75%          2.000000
max          2.000000
```

[44]:
```python
# 2. Analisis Adicionales (I):
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import skew, kurtosis


for col in df.select_dtypes(include='number'):
    print(f"{col}: Skewness = {skew(df[col])}, Kurtosis = {kurtosis(df[col])}")
```

```
a1_score: Skewness = -1.0116830756233122, Kurtosis = -0.9764973544973548
a2_score: Skewness = 0.18364303357928888, Kurtosis = -1.9662752362177962
a3_score: Skewness = 0.17776627705192047, Kurtosis = -1.9683991507431002
a4_score: Skewness = 0.01451417028005336, Kurtosis = -1.9997893388610817
a5_score: Skewness = 0.0029027606770736024, Kurtosis = -1.999991573980452
```

```
a6_score: Skewness = 0.9793166442751312, Kurtosis = -1.040938910245697
a7_score: Skewness = 0.32647058823529396, Kurtosis = -1.8934169550173015
a8_score: Skewness = -0.650133962692343, Kurtosis = -1.5773258305539513
a9_score: Skewness = 0.7609062596257509, Kurtosis = -1.4210216640623496
a10_score: Skewness = -0.3144095631436657, Kurtosis = -1.9011466266038095
age: Skewness = 1.0281046532083644, Kurtosis = 0.4854422124812272
gender: Skewness = -0.09590106211762979, Kurtosis = -1.990802986284711
jundice: Skewness = 2.7186943799097536, Kurtosis = 5.391299131352882
austim: Skewness = 2.2113132675044493, Kurtosis = 2.8899063670412017
result: Skewness = 0.32667942129588323, Kurtosis = -0.7131860516040089
eth_cat: Skewness = 0.6148800158922599, Kurtosis = -1.1860113561842176
aq_binary: Skewness = 0.5445240776805577, Kurtosis = -1.7034935288261377
res_code: Skewness = -0.5960438734159403, Kurtosis = -0.7439126801859488
relation_coded: Skewness = -1.6685486450127798, Kurtosis = 1.6059952296892908
```

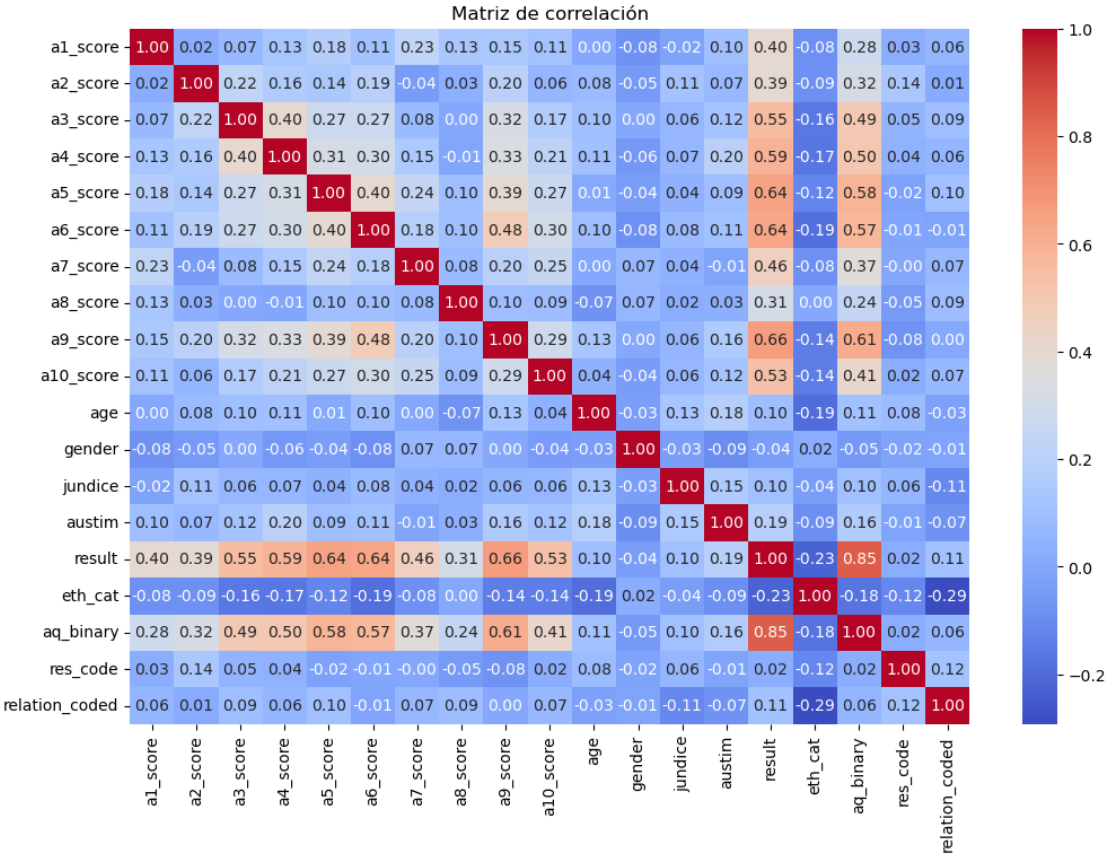
[45]:
```
# visualizamos la asimetria positiva de 'result
plt.hist(df['result'], bins=11)  # 11 bins para cada valor posible de 0 a 10
plt.xlabel('Puntuación AQ-10')
plt.ylabel('Frecuencia')
plt.title('Distribución de la puntuación AQ-10')
plt.show()
```



[49]:
```
# importamos la libreria 'vega_datasets'
# si no esta instalada, la instalamos
# %pip install altair vega_datasets
```

[48]:
```
# Analisis adicionales (II):
import altair as alt

# 1. Calculamos la matriz de correlación
correlation_matrix = df.corr()

# 2. Creamos un heatmap con Seaborn
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")

# 3. & 4. Mostramos el heatmap con título
plt.title('Matriz de correlación')
plt.show()

# 5. Convertimos el heatmap de Seaborn a un gráfico de Altair
heatmap_data = correlation_matrix.stack().reset_index().rename(
    columns={"level_0": "Variable 1",
             "level_1": "Variable 2",
             0: "Correlación"})

base = alt.Chart(heatmap_data).encode(
    x='Variable 1:O',
    y='Variable 2:O',
    color=alt.Color(
        'Correlación:Q', scale=alt.Scale(
            scheme='blueorange', domain=[-1, 1])),
    tooltip=['Variable 1', 'Variable 2', 'Correlación']
)

# Heatmap
heatmap = base.mark_rect().encode()

# Etiquetas
text = base.mark_text(baseline='middle').encode(
    text=alt.Text('Correlación:Q', format='.2f'),
    color=alt.condition(
        alt.datum.Correlación > 0.5,
        alt.value('white'),
        alt.value('black')
    )
)
```

```
# Combinamos heatmap y texto
chart = heatmap + text

# 13. Configuramos el título del gráfico
chart = chart.properties(
    title='Matriz de correlación - Altair',
    width=800,  # Ancho en píxeles
    height=600  # Alto en píxeles
)

# 14. Mostramos el gráfico Altair
chart.show()
```



`alt.LayerChart(…)`

El análisis exploratorio de los datos reveló características clave:

1. Distribución de las respuestas al test AQ-10: Las respuestas a las preguntas individuales del test AQ-10 mostraron una distribución relativamente uniforme, con una media cercana a 0.5 y una desviación estándar similar, lo que sugiere una ausencia de tendencia clara hacia respuestas afirmativas o negativas en las preguntas.

2. Características demográficas: La mayoría de los participantes eran adultos jóvenes, con una media de edad de 29.2 años. Se observó un ligero desequilibrio de género, con una mayor proporción de hombres (52.4%) que de mujeres (47.6%). La prevalencia de ictericia al nacer y de autismo fue baja (9.7% y 12.9%, respectivamente).

3. Puntuación total AQ-10: La distribución de la puntuación total en el AQ-10 (result) presentó una asimetría positiva, indicando que la mayoría de los participantes obtuvieron puntuaciones bajas, con pocos casos de puntuaciones altas, lo cual es consistente con la baja prevalencia de autismo en la muestra.

4. Etnia y país de residencia: La mayoría de los participantes eran de etnia blanca y residían en Estados Unidos. No obstante, el conjunto de datos incluyó una diversidad de etnias y países de residencia, lo que podría ser útil para explorar posibles diferencias culturales o geográficas en la manifestación del autismo.

5. Relación con la persona que proporcionó la información: La mayoría de las respuestas fueron proporcionadas por los propios participantes, lo que aumenta la fiabilidad de los datos.

Matriz de correlación:

El análisis de la matriz de correlación no reveló correlaciones lineales fuertes entre las variables, lo que sugiere que la predicción del autismo puede depender de una combinación compleja de factores, más que de una sola variable dominante.

A pesar de la ausencia de correlaciones lineales fuertes, se decidió mantener todas las variables en esta etapa. Se explorarán relaciones no lineales y posibles interacciones entre las variables a medida que se avance en el desarrollo de los modelos.

Próximos pasos y consideraciones:

En las siguientes secciones, se construirán y evaluarán diversos modelos de aprendizaje automático: clasificación binaria, regresión lineal y clasificación multiclase, con el apoyo de modelos no supervisados. Se utilizará la validación cruzada y métricas de evaluación apropiadas para comparar el rendimiento de los modelos y seleccionar el más adecuado para predecir el riesgo de TEA en adultos.

El objetivo principal es extraer consideraciones sobre cómo influyen las variables en el diagnóstico de autismo.

Es importante reconocer que el conjunto de datos presenta un desequilibrio de clases, con una mayor proporción de individuos sin autismo. Se abordará este desequilibrio durante el entrenamiento y la evaluación de los modelos, utilizando técnicas como el remuestreo o la ponderación de clases. Sin embargo, se reconoce que la muestra y las variables consideradas pueden no ser suficientes para la creación de un modelo perfecto, lo que resalta la complejidad del diagnóstico del TEA y la necesidad de futuras investigaciones.

1.4 Modelos No Supervisados

Se exploraron modelos no supervisados con el objetivo de obtener una comprensión más profunda de los datos. Esta exploración tuvo como metas potenciales:

Descubrir patrones subyacentes en los datos que no son evidentes en un análisis superficial. Generar nuevas características que puedan mejorar la capacidad predictiva de los modelos supervisados.

Obtener una mejor comprensión de la estructura y las relaciones entre las variables. Potencialmente, mejorar el rendimiento de los modelos de clasificación.

1.4.1 Ingeniería de Características

Discretización de la Edad Dado que la edad, como variable numérica continua, podría presentar una relación no lineal con el diagnóstico de autismo, se decidió discretizarla en intervalos para capturar posibles patrones no lineales y mejorar la capacidad predictiva de los modelos.

Se optó por dividir la edad en cuartiles, es decir, en cuatro grupos de igual tamaño según su distribución en el conjunto de datos. Esta estrategia permitió explorar posibles efectos de la edad en diferentes etapas de la vida adulta, sin asumir una relación lineal predefinida.

La función `qcut` de Pandas fue utilizada para realizar la discretización, asignando etiquetas descriptivas a cada cuartil (Q1, Q2, Q3, Q4). Posteriormente, se aplicó Label Encoding para convertir estas etiquetas en valores numéricos ordinales, facilitando su uso en los modelos de aprendizaje automático.

Justificación de la elección:

La división en cuartiles se consideró adecuada debido a la ausencia de una hipótesis específica sobre la relación entre la edad y el autismo. Al no contar con información previa sobre puntos de corte relevantes, los cuartiles permitieron una exploración más flexible de la variable edad y la posibilidad de descubrir patrones no evidentes en su forma continua.

Consideraciones:

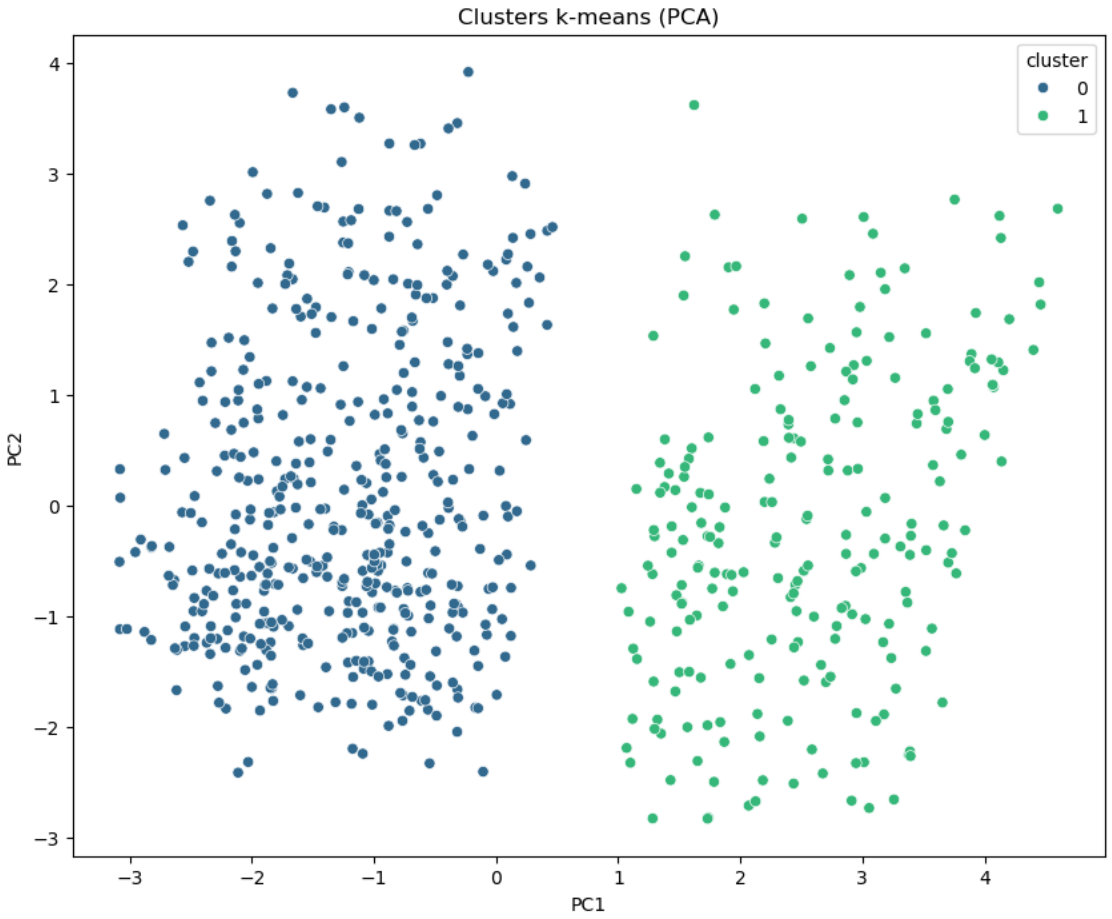
- Pérdida de información: La discretización de la edad implica una pérdida de granularidad en la información. Sin embargo, esta pérdida se compensa con la posibilidad de capturar relaciones no lineales y mejorar la interpretabilidad de los resultados.
- Evaluación del impacto: El impacto de esta transformación en el rendimiento de los modelos fue evaluado en etapas posteriores del proyecto.

Modelo K-Means

[57]:
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import (
    silhouette_score, davies_bouldin_score, calinski_harabasz_score)

# importamos el df previamente limpiado
df = pd.read_csv("autism_adult_cleaned.csv")

# Convertimos la edad en cuartiles
df['age_group'] = pd.qcut(df['age'], q=4, labels=['Q1', 'Q2', 'Q3', 'Q4'])

# Codificamos la nueva variable categórica 'age_group'
le = LabelEncoder()
df['age_group_encoded'] = le.fit_transform(df['age_group'])
```

```
# visualizamos los valores unicos dentro de la columna 'age_group_encoded'
df['age_group_encoded'].unique()

# eliminamos la columna 'age_group'
df.drop(columns=['age_group'], inplace=True)

# Seleccionamos las caracteristicas:
X = df.drop(columns=['austim', 'result', 'aq_binary'])

# Escalamos las caracteristicas
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Creamos un modelo KMeans con 2 clusters
kmeans = KMeans(n_clusters=2, random_state=42)
df['k-means'] = kmeans.fit_predict(X_scaled)


# Evaluamos la calidad de los clusters
silhouette_avg = silhouette_score(X_scaled, df['k-means'])
davies_bouldin_index = davies_bouldin_score(X_scaled, df['k-means'])
calinski_harabasz_index = calinski_harabasz_score(X_scaled, df['k-means'])

# Imprimimos los indices
print("Puntuación de silueta (K-Means):", silhouette_avg)
print("Índice de Davies-Bouldin (K-Means):", davies_bouldin_index)
print("Índice de Calinski-Harabasz (K-Means):", calinski_harabasz_index)

# Aplicamos PCA para reducir la dimensionalidad a 2 componentes
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Creamos un DataFrame con las componentes
# principales y las etiquetas de los clusters
pca_df = pd.DataFrame(data=X_pca, columns=['PC1', 'PC2'])
pca_df['cluster'] = df['k-means']

# Graficamos los puntos coloreados según el cluster
plt.figure(figsize=(10, 8))
sns.scatterplot(
    x='PC1', y='PC2', hue='cluster', data=pca_df, palette='viridis')
plt.title('Clusters k-means (PCA)')
plt.show()
```

```
Puntuación de silueta (K-Means): 0.187503850828204
Índice de Davies-Bouldin (K-Means): 1.977273056530785
Índice de Calinski-Harabasz (K-Means): 150.28624423037508
```



Evaluación de la agrupación con K-means

Los resultados de la agrupación k-means, visualizados mediante PCA y evaluados con métricas internas, sugieren una estructura de clusters moderadamente definida en los datos.

- Puntuación de silueta: El valor de 0.188 indica una separación justa entre los clusters, con cierto grado de solapamiento. Aunque no es ideal, sugiere que los clusters tienen cierta coherencia interna y se distinguen, aunque no de forma muy marcada.
- Índice de Davies-Bouldin: El valor de 1.98 respalda la observación anterior. Este índice mide la similitud entre clusters, y un valor más bajo es preferible. En este caso, el valor relativamente alto indica que algunos clusters podrían estar cercanos o solapados.
- Índice de Calinski-Harabasz: Con un valor de 150.29, este índice sugiere una calidad de clustering decente. Este índice favorece clusters densos y bien separados, por lo que el valor obtenido indica una estructura de clusters razonable.

Gráfico de dispersión (PCA):

La visualización de los clusters en el espacio de las dos primeras componentes principales muestra una separación parcial entre los dos grupos. Si bien hay una tendencia a que los clusters se agrupen en diferentes regiones del gráfico, también hay un solapamiento considerable, especialmente en la

zona central. Esto confirma la interpretación de las métricas internas, indicando que los clusters no están completamente separados.

En general parece que algortimo del modelo *K-means* ha logrado identificar dos clusters en los datos, pero la separación entre ellos no es perfecta. Esto podría deberse a la naturaleza de los datos, a la elección de las características o a la inherente complejidad del problema de detección del autismo.

Evaluación de la Agrupación con K-means
Los resultados de la agrupación k-means, visualizados mediante PCA y evaluados con métricas internas, sugirieron una estructura de clusters moderadamente definida en los datos.

- Puntuación de silueta: El valor de 0.188 indicó una separación aceptable entre los clusters, aunque con cierto grado de solapamiento. Esto sugirió que los clusters presentaban cierta coherencia interna y se distinguían entre sí, pero no de forma muy marcada.
- Índice de Davies-Bouldin: El valor de 1.98 respaldó la observación anterior. Este índice, que mide la similitud entre clusters, mostró un nivel moderado de separación entre los grupos, indicando que algunos clusters podrían estar cercanos o solapados.
- Índice de Calinski-Harabasz: Con un valor de 150.29, este índice sugirió una calidad de clustering aceptable. Al favorecer clusters densos y bien separados, el valor obtenido apuntó hacia una estructura de clusters razonable.

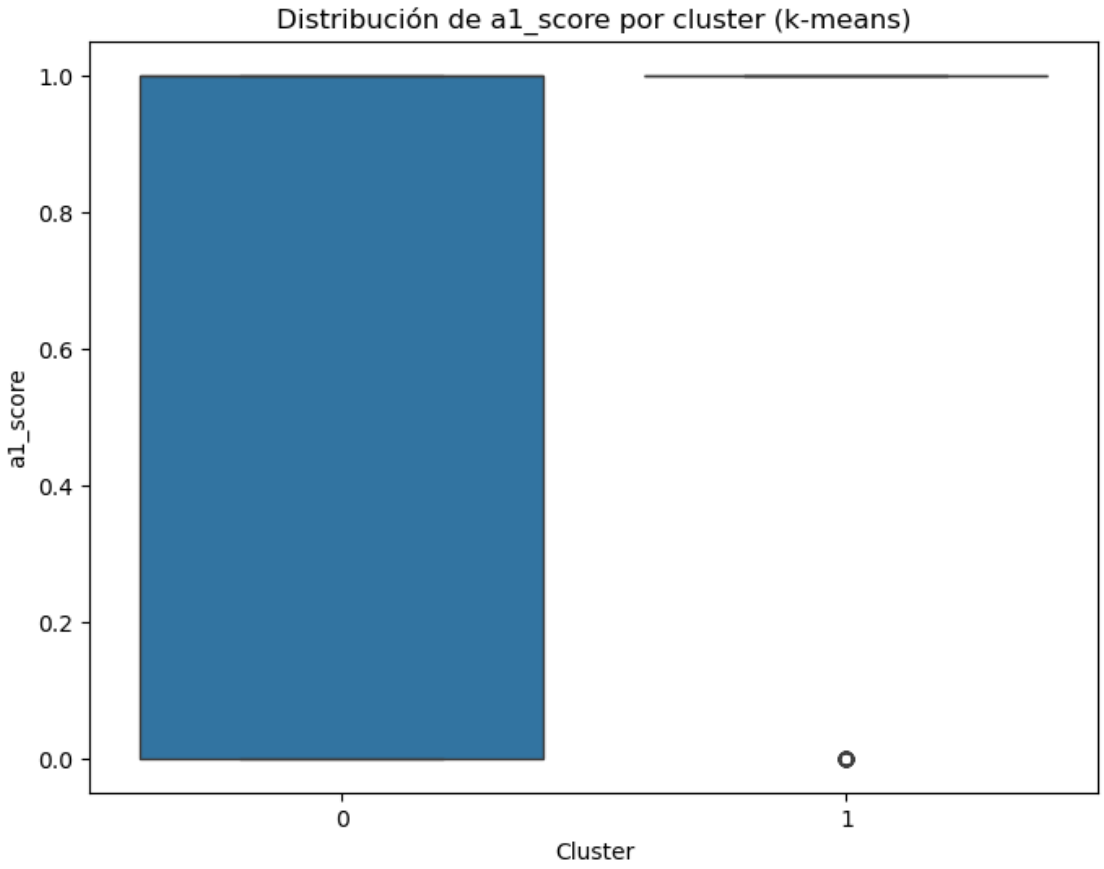
Gráfico de dispersión (PCA):

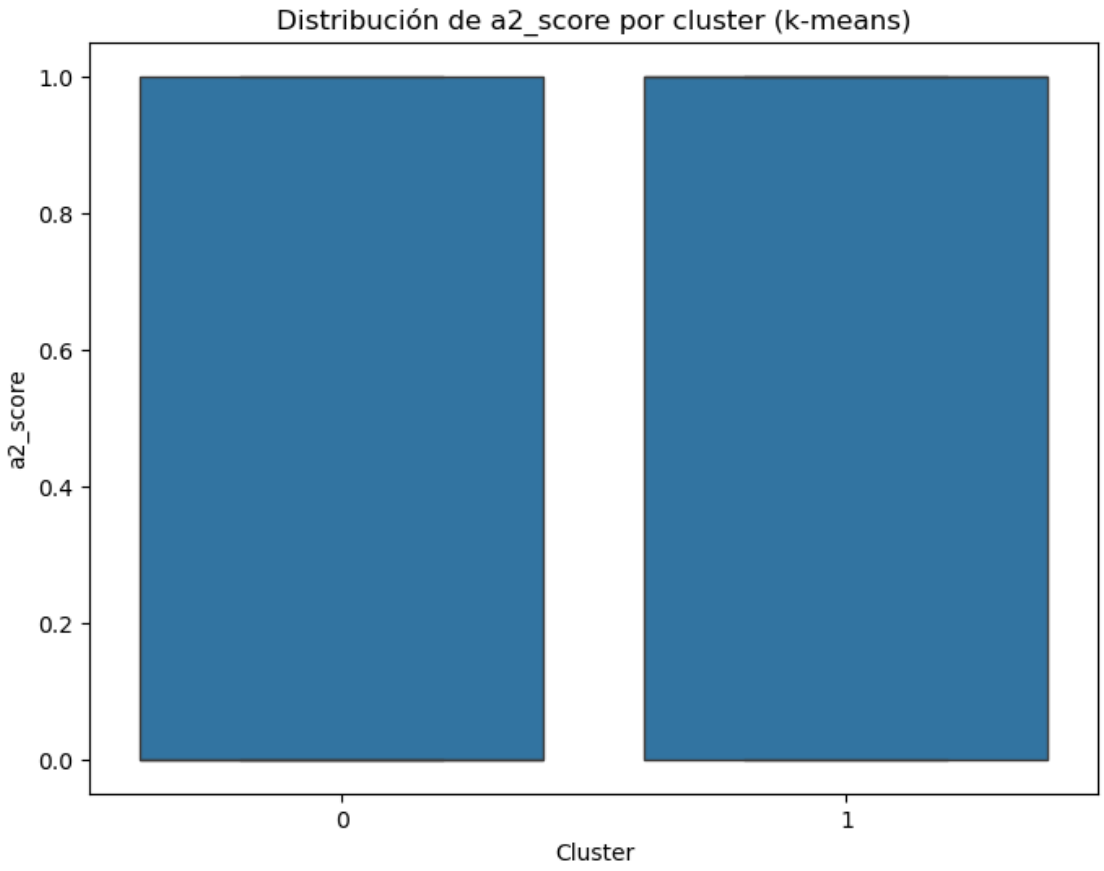
La visualización de los clusters en el espacio de las dos primeras componentes principales mostró una separación parcial entre los dos grupos. Si bien se observó una tendencia a que los clusters se agruparan en diferentes regiones del gráfico, también hubo un solapamiento considerable, especialmente en la zona central. Esto confirmó la interpretación de las métricas internas, indicando que los clusters no estaban completamente separados.

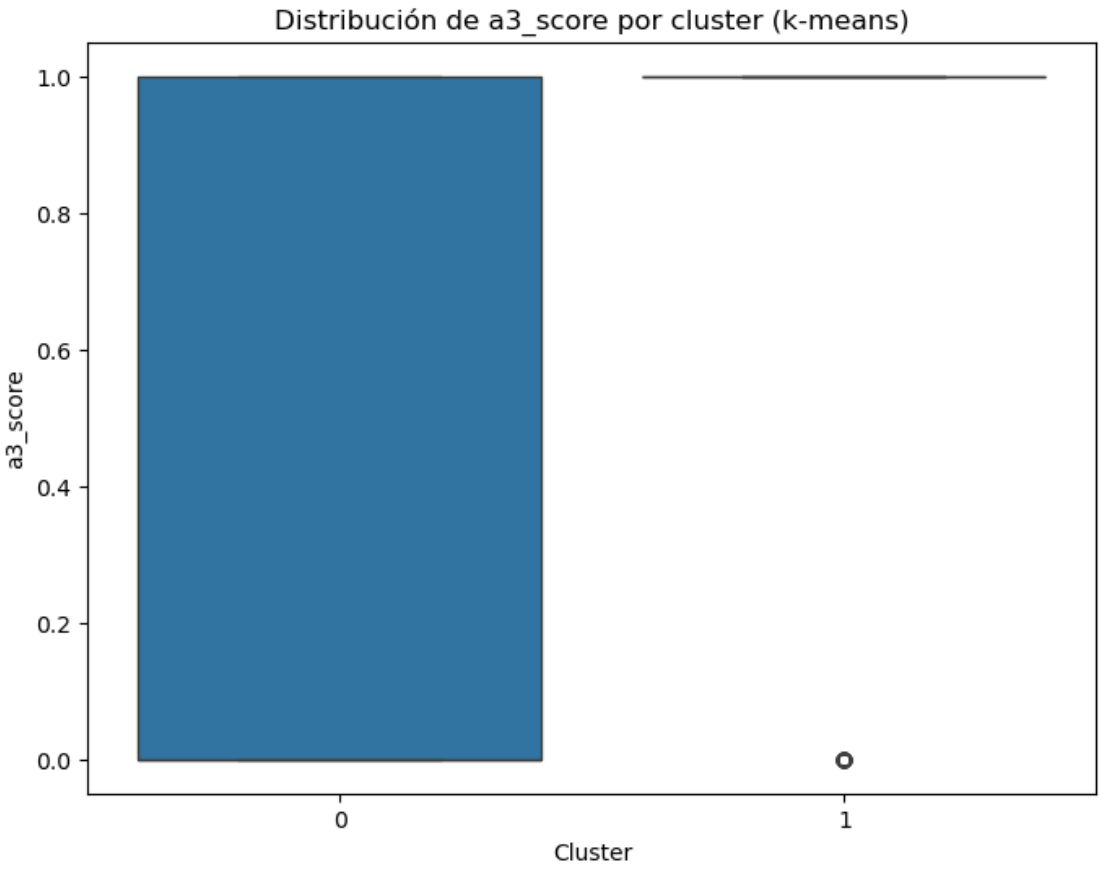
En general, el algoritmo k-means logró identificar dos clusters en los datos, pero la separación entre ellos no fue perfecta. Esto podría atribuirse a la naturaleza de los datos, a la elección de las características o a la inherente complejidad del problema de detección del autismo.

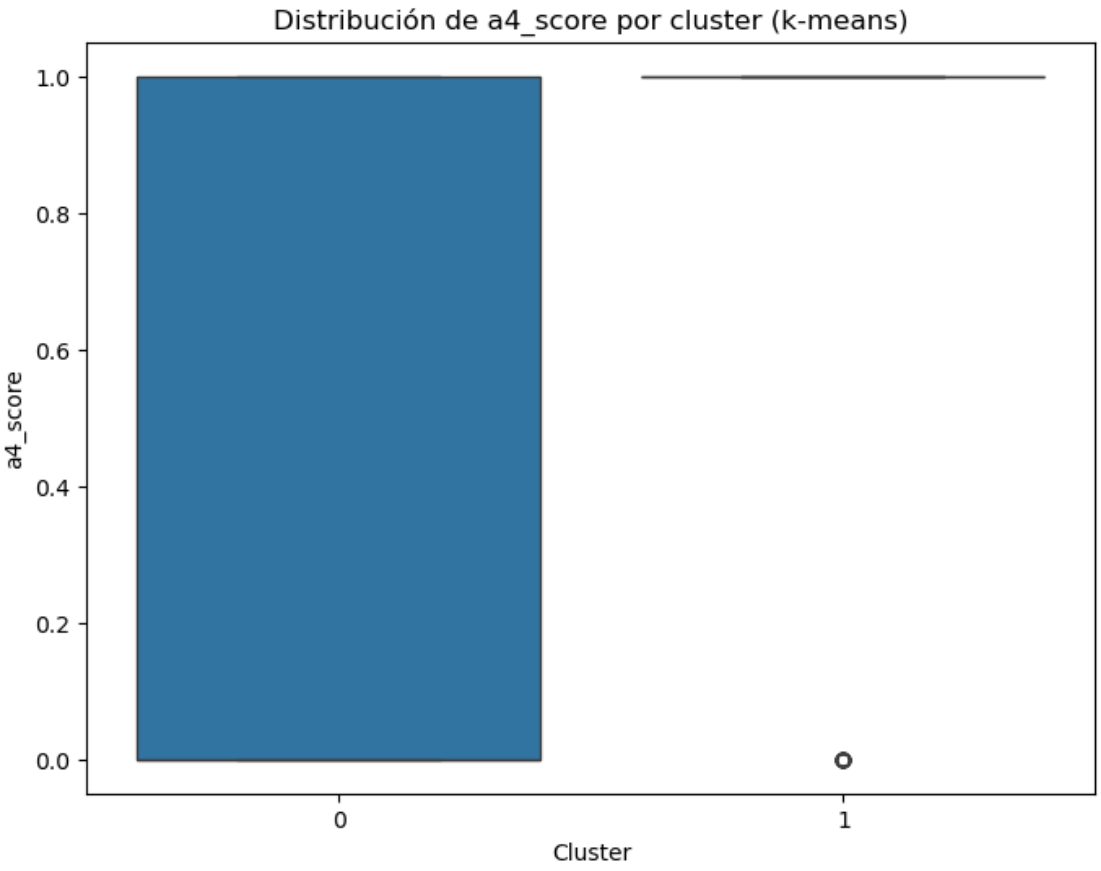
[59]:
```
# Graficamos la distribución de las
# variables a los clusters:
import seaborn as sns
import matplotlib.pyplot as plt

# Convertimos X_scaled a DataFrame
X_scaled_df = pd.DataFrame(X_scaled, columns=X.columns)
# Iteramos sobre las columnas numéricas
# escaladas para visualizar los clusters
for col in X_scaled_df.columns:
    # Verificamos si la columna es numérica (int64)
    if df[col].dtype == 'int64':
        plt.figure(figsize=(8, 6))
        sns.boxplot(x='k-means', y=col, data=df)
        plt.title(f'Distribución de {col} por cluster (k-means)')
```

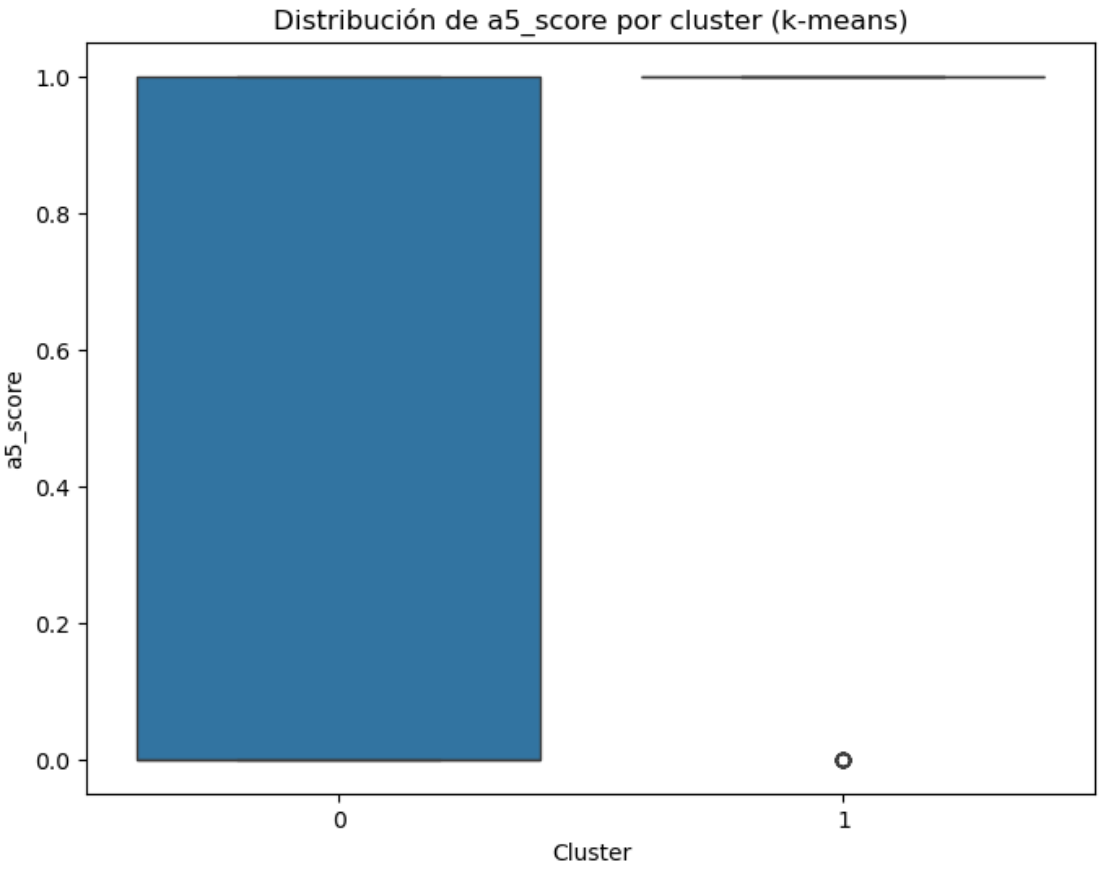
```
        plt.xlabel('Cluster')
        plt.ylabel(col)
        plt.show()
```

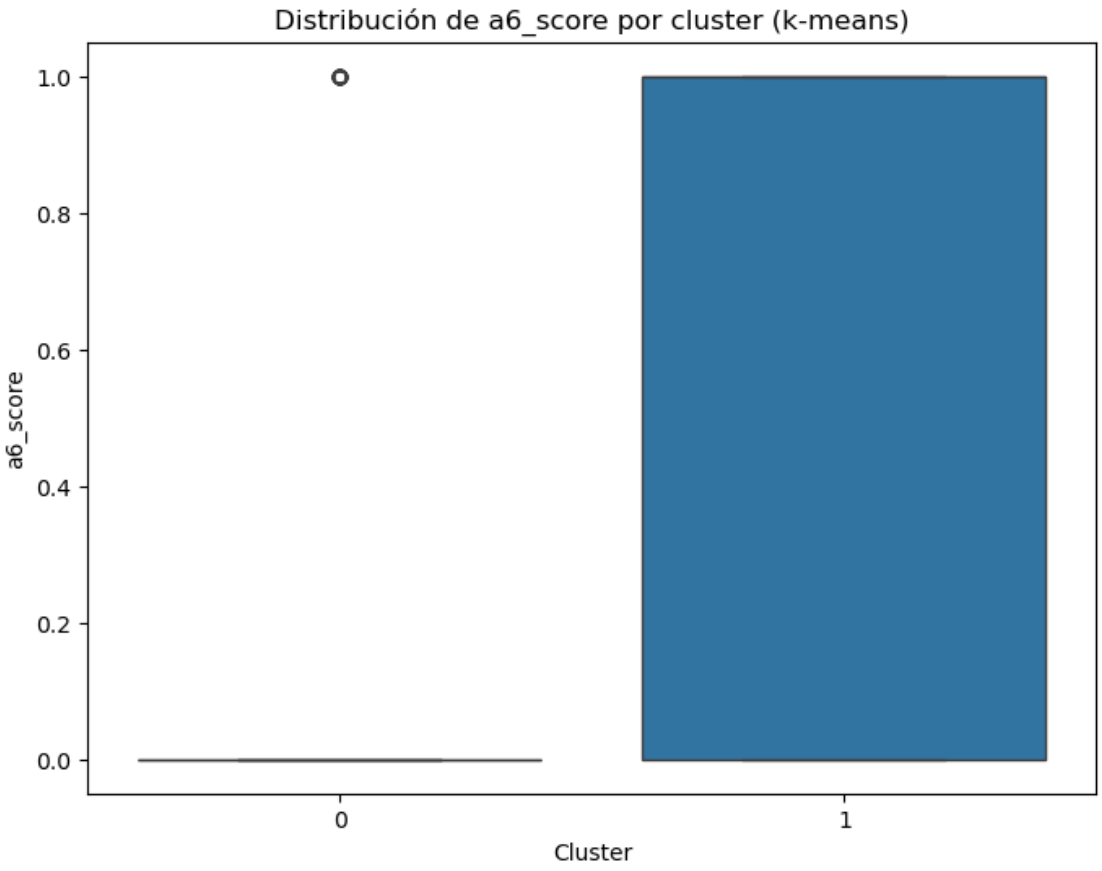


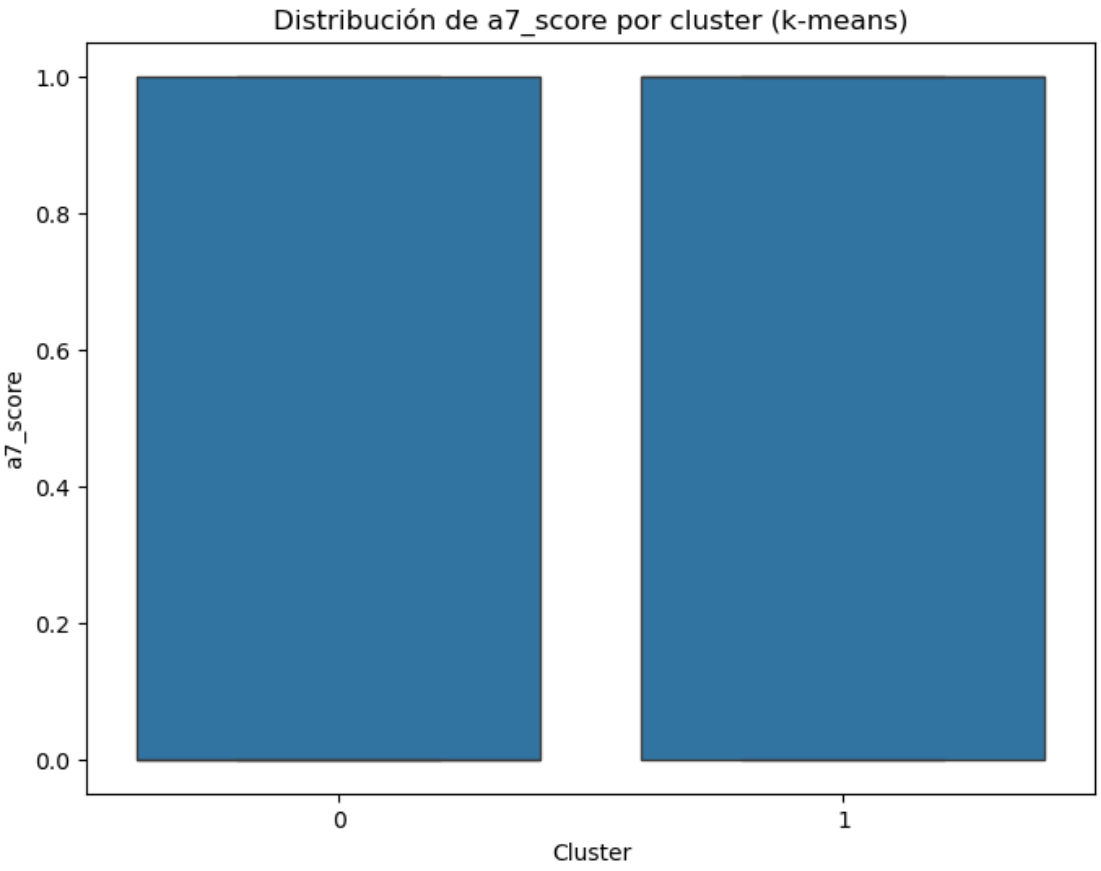

Distribución de a2_score por cluster (k-means)
a2_score
1.0
0.8
0.6
0.4
0.2
0.0
0
1
Cluster

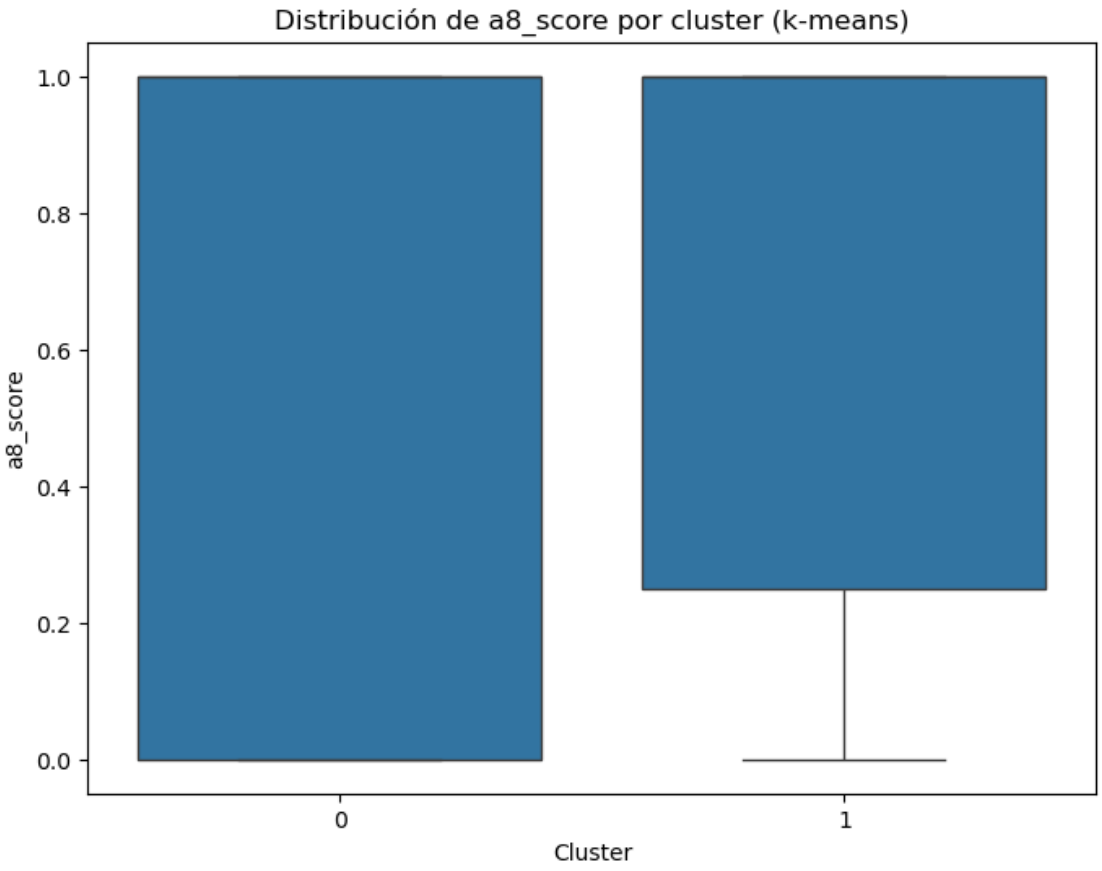

Distribución de a3_score por cluster (k-means)
a3_score
1.0
0.8
0.6
0.4
0.2
0.0
0
1
Cluster

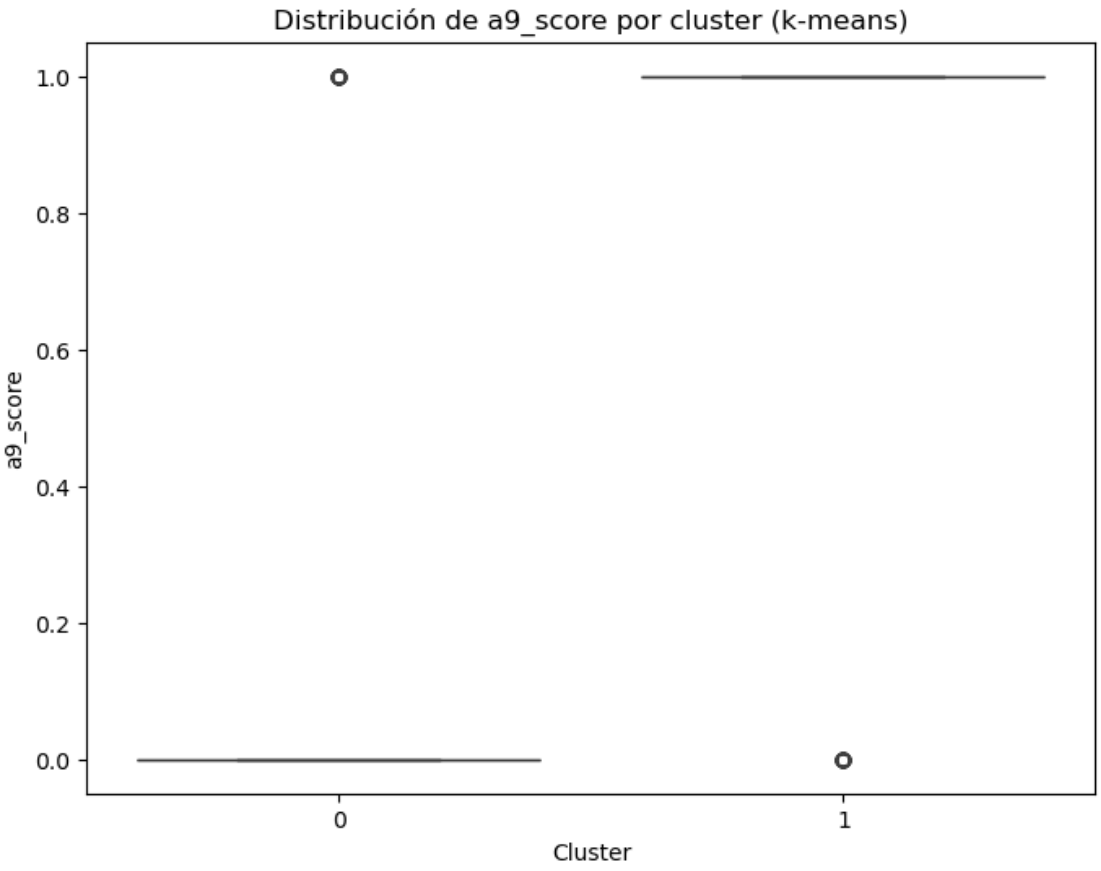

Distribución de a4_score por cluster (k-means)
a4_score
1.0
0.8
0.6
0.4
0.2
0.0
0
1
Cluster

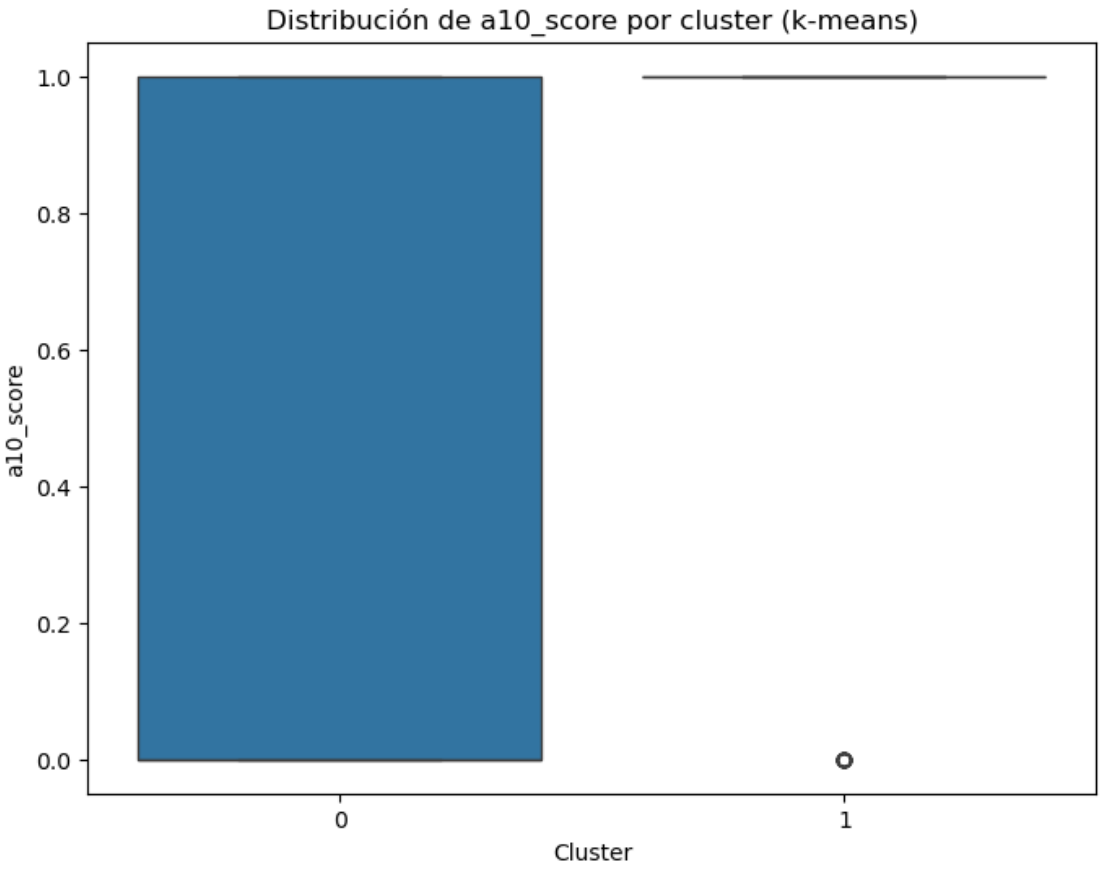

Distribución de a5_score por cluster (k-means)
a5_score
1.0
0.8
0.6
0.4
0.2
0.0
0
1
Cluster

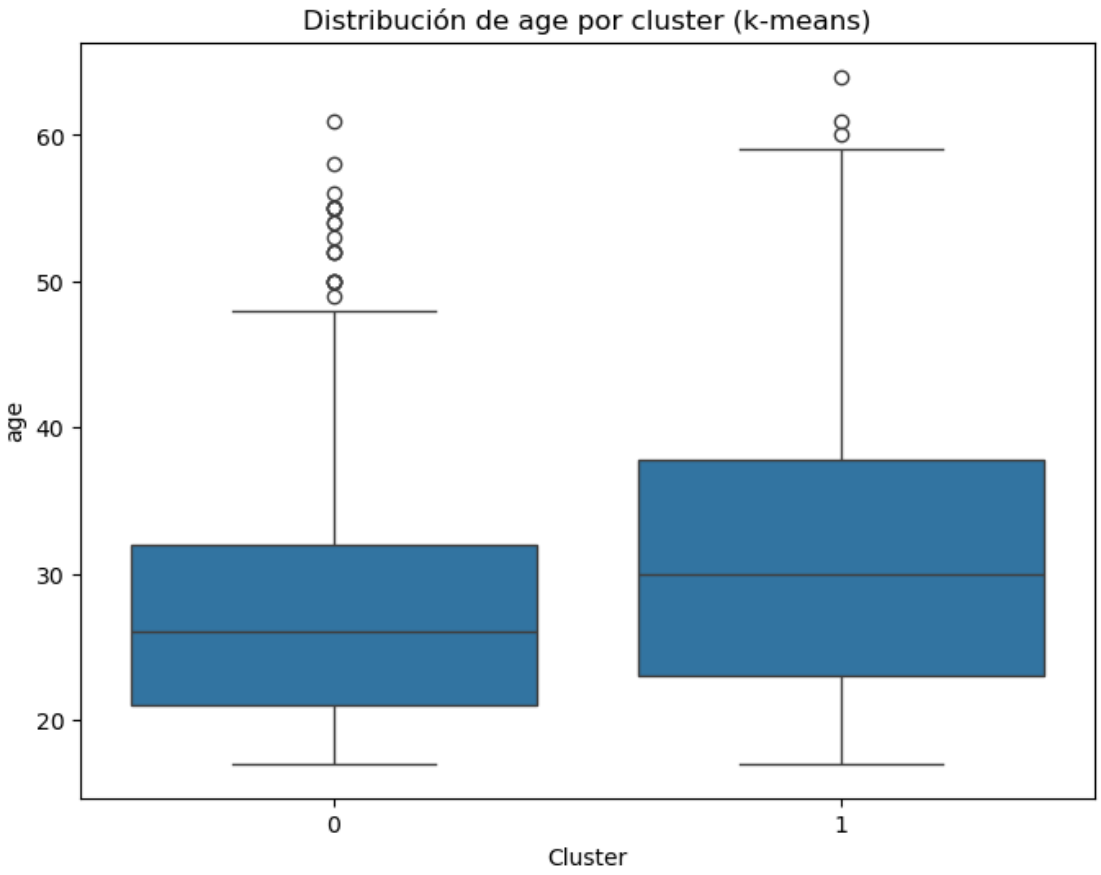

Distribución de a6_score por cluster (k-means)
a6_score
1.0
0.8
0.6
0.4
0.2
0.0
0
1
Cluster

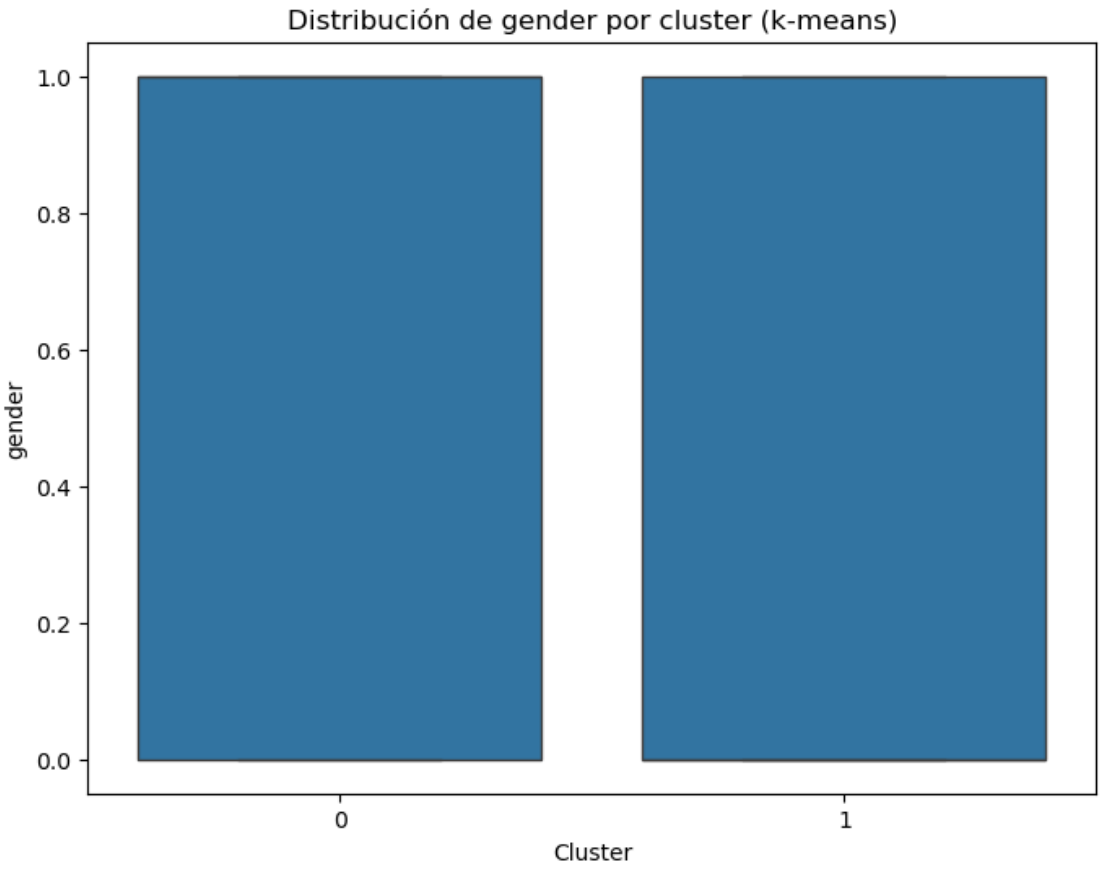

Distribución de a7_score por cluster (k-means)
1.0
0.8
0.6
0.4
0.2
0.0
a7_score
0
1
Cluster

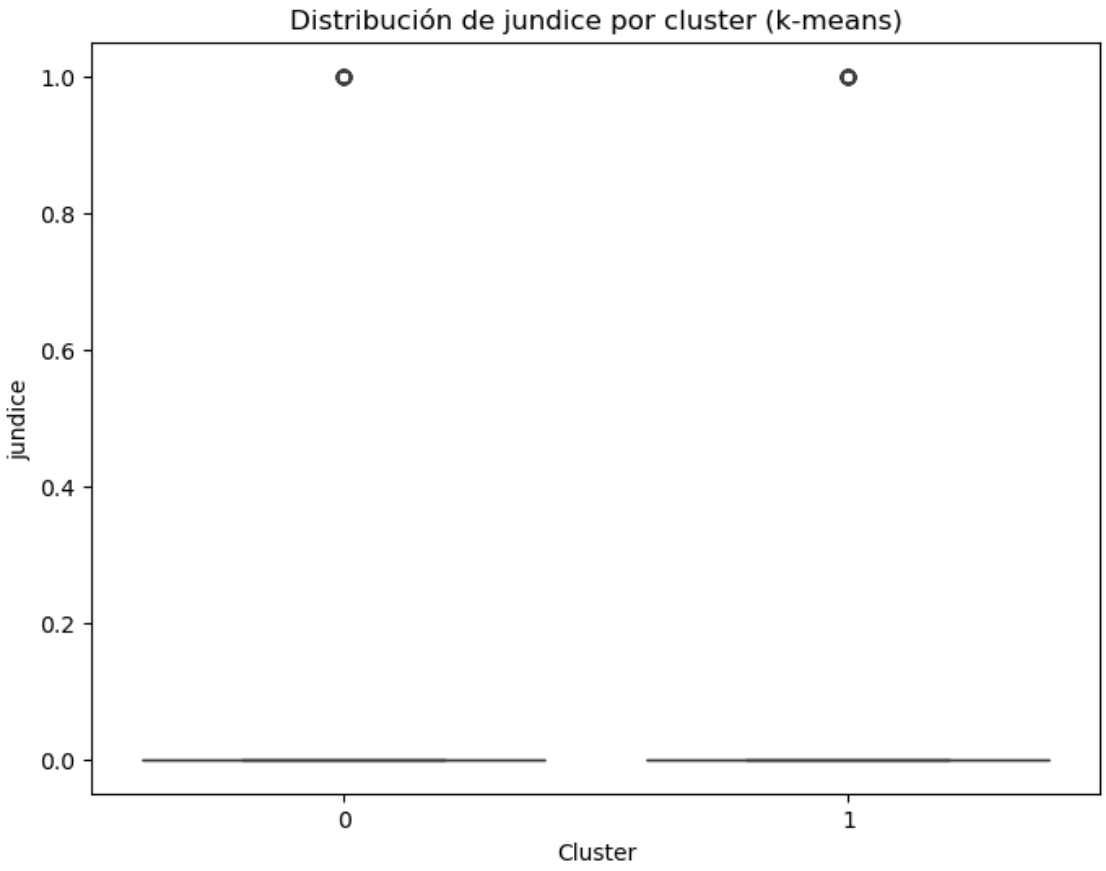

Distribución de a8_score por cluster (k-means)
a8_score
1.0
0.8
0.6
0.4
0.2
0.0
0
1
Cluster

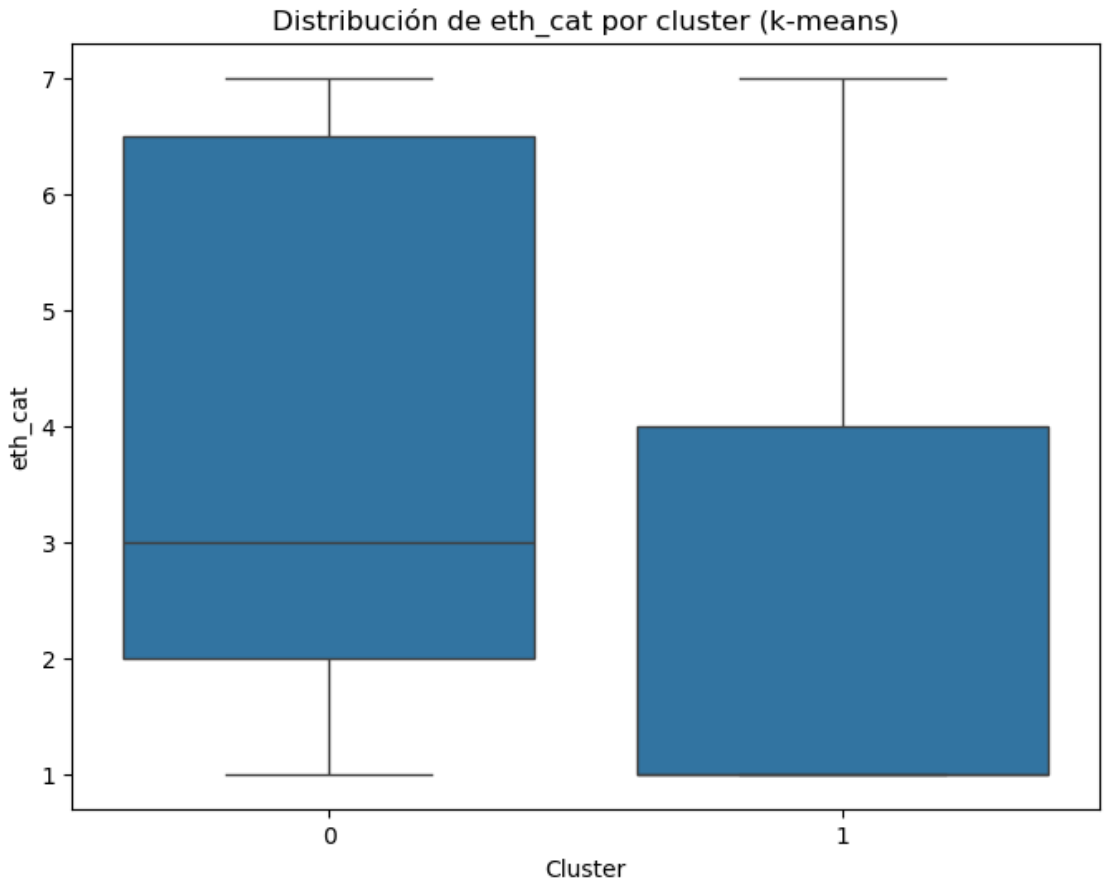

Distribución de a9_score por cluster (k-means)
1.0
0.8
0.6
0.4
0.2
0.0
a9_score
0
1
Cluster

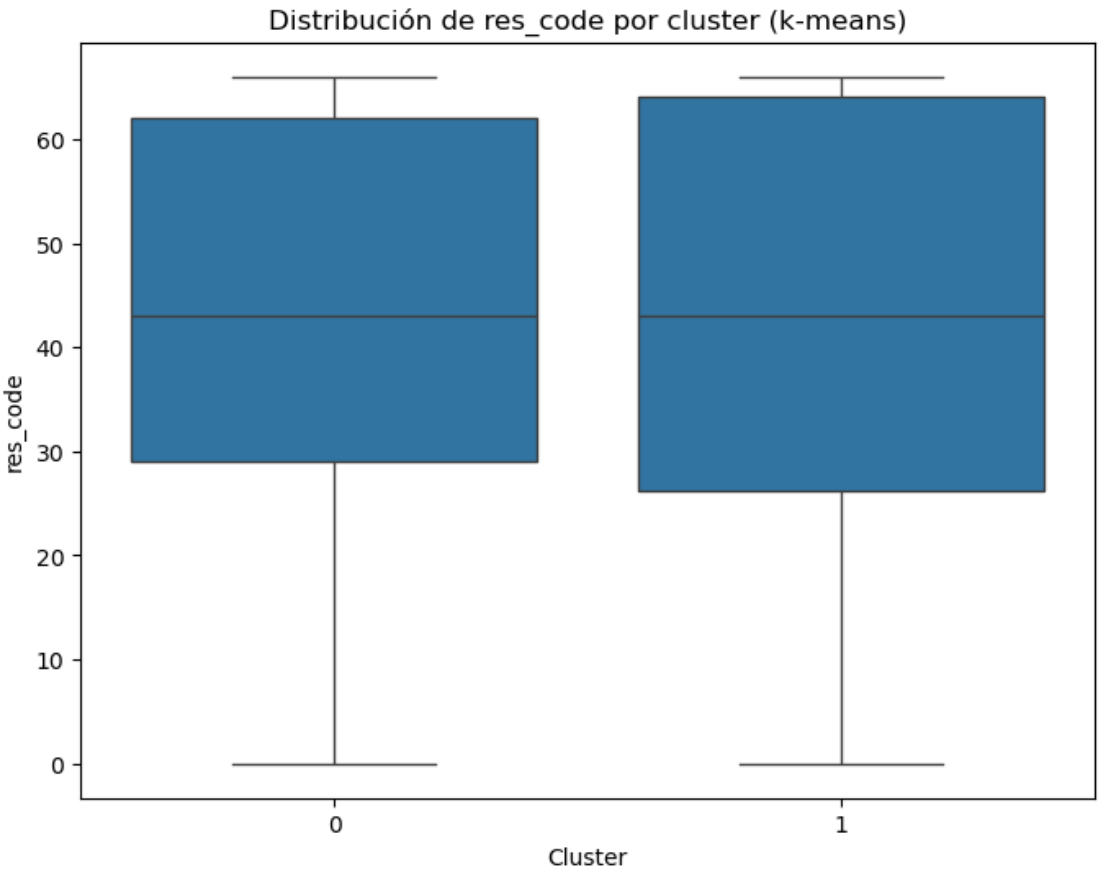

Distribución de a10_score por cluster (k-means)
a10_score
1.0
0.8
0.6
0.4
0.2
0.0
0
1
Cluster

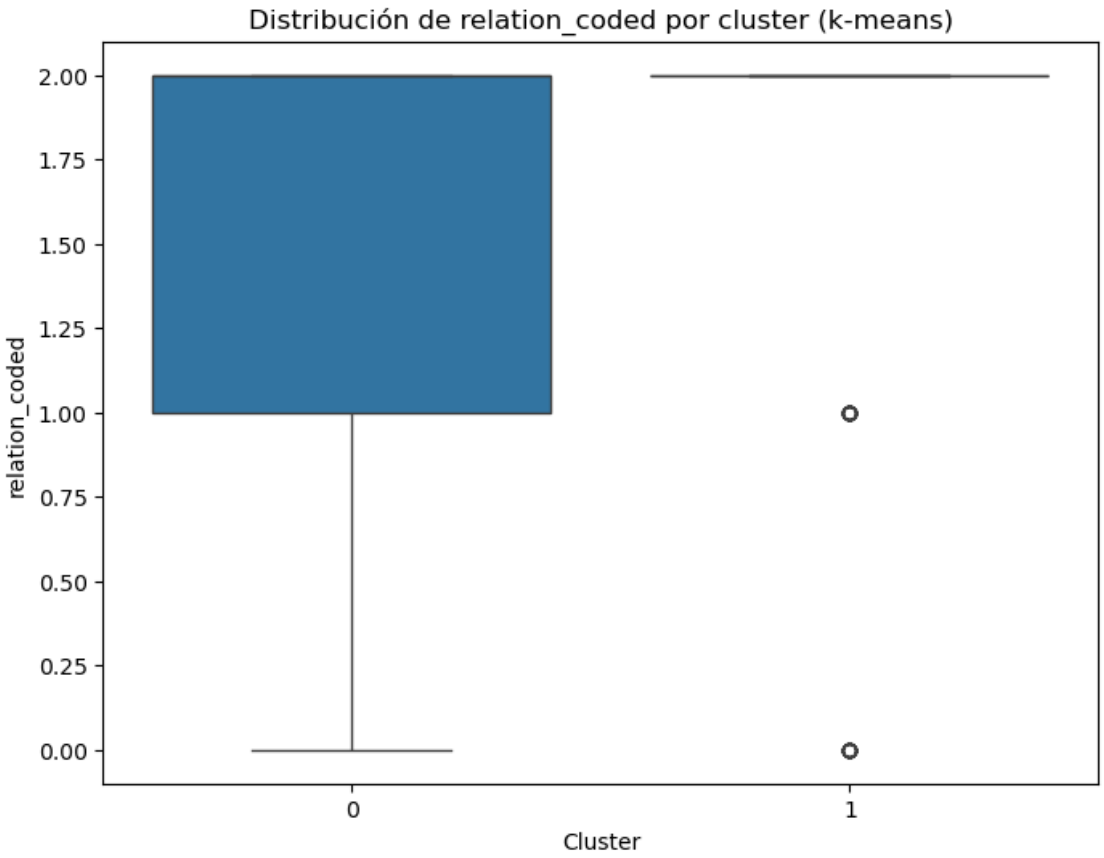

Distribución de age por cluster (k-means)
age
60
50
40
30
20
0
1
Cluster

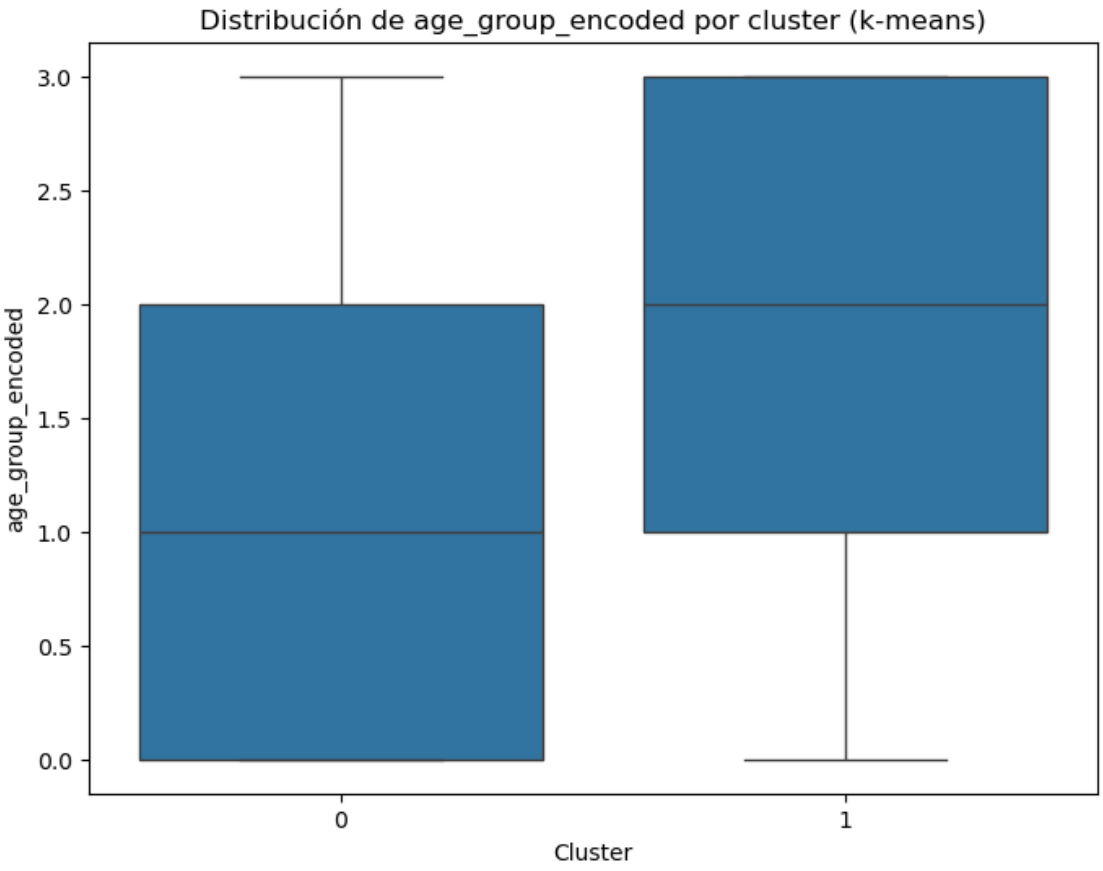

Distribución de gender por cluster (k-means)
gender
1.0
0.8
0.6
0.4
0.2
0.0
0
1
Cluster


Distribución de jundice por cluster (k-means)
jundice
1.0
0.8
0.6
0.4
0.2
0.0
0
1
Cluster


Distribución de eth_cat por cluster (k-means)
eth_cat
7
6
5
4
3
2
1
0
1
Cluster


Distribución de res_code por cluster (k-means)
res_code
60
50
40
30
20
10
0
0
1
Cluster


Distribución de relation_coded por cluster (k-means)
relation_coded
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
1
Cluster



Análisis de la Relevancia de las Características en la Formación de Clusters
El análisis exploratorio de datos (EDA) mediante la técnica de clustering k-means permitió evaluar la contribución de cada característica en la formación de los dos clusters identificados.

Puntuaciones AQ
En particular, se observó que las variables a7_score y a2_score presentaban una distribución prácticamente idéntica en ambos clusters, lo que sugiere que estas características no aportaron información discriminante para diferenciar entre los grupos. La ausencia de variabilidad en estas variables entre los clusters indica que no estuvieron asociadas a los patrones subyacentes que k-means utilizó para la agrupación.

Género Asimismo, el análisis de la variable gender reveló una distribución similar de hombres y mujeres en ambos clusters. Esto sugiere que el género, en el contexto de este conjunto de datos, no parece ser un factor relevante en la formación de los clusters ni en la manifestación del Trastorno del Espectro Autista (TEA).

Residencia
El análisis del gráfico de caja (boxplot) que ilustra la distribución de res_code (código de país de residencia) por cluster en k-means, reveló que el lugar de residencia no pareció ser un factor determinante en la formación de los clusters.

- Distribución similar: Las medianas y los rangos intercuartílicos de ambos clusters resultaron

ser muy parecidos, lo que indica que la distribución de los códigos de país fue prácticamente la misma en ambos grupos.

- Ausencia de patrones en outliers: Aunque se observaron algunos valores atípicos (outliers) en ambos clusters, no se identificó un patrón claro que sugiriera una asociación entre países específicos y un cluster en particular.

Dado que la distribución de res_code fue muy similar en ambos clusters, se concluyó que el lugar de residencia no tuvo un papel discriminante en la formación de los grupos por parte de k-means, lo que sugiere que esta variable no aporta información significativa para distinguir entre los dos clusters identificados.

Ictericia En el caso de `jundice` tambien podemos concluir que la ictericia no necesariamente parece estar relacionada con el Trastorno del Espectro Autista. La mayoría de los valores son 0 en ambos clusters: Las líneas centrales de ambas cajas están en 0, lo que significa que la mayoría de los individuos en ambos clusters tienen un valor de 0.

Edad (Sin escalas)
La variable `age` (sin discretizar) parece tener cierta capacidad discriminatoria entre los dos clusters generados por k-means.

- Mediana más alta en el Cluster 1: La línea central dentro de la caja (la mediana) es notablemente más alta en el Cluster 1 que en el Cluster 0. Esto indica que la edad típica de los individuos en el Cluster 1 es mayor que la de aquellos en el Cluster 0.
- Rango intercuartílico similar: La altura de las cajas (que representa el rango intercuartílico o IQR) es bastante similar en ambos clusters, lo que sugiere que la dispersión o variabilidad de las edades dentro de cada cluster es comparable.
- Valores atípicos en el Cluster 0: El Cluster 0 muestra varios valores atípicos (los puntos individuales por encima del bigote superior), lo que indica que hay algunos individuos en este cluster con edades significativamente mayores que la mayoría. El Cluster 1, en cambio, no parece tener valores atípicos.

La diferencia en las medianas y la presencia de outliers en el Cluster 0 sugieren que la edad, incluso sin discretizar, tiene cierta capacidad para separar los dos clusters. Esto implica que la edad podría ser un factor que k-means ha tenido en cuenta al formar los grupos.

Etnia
La etnicidad (`eth_cat`) parece ser un factor que influye en la formación de los clusters, ya que hay una clara diferencia en la distribución de las categorías étnicas entre los dos grupos.

- Cluster 0: Predominantemente de etnia blanca.
- Cluster 1: Más diverso étnicamente, con una mayor proporción de individuos de etnias minoritarias o agrupadas en la categoría "Otros".
- Potencial predictor: La etnia podría estar asociada con factores genéticos, ambientales o socioculturales que influyen en el riesgo de autismo. Incluirla como característica en los modelos podría mejorar la capacidad predictiva al capturar estas diferencias.
- Identificación de disparidades: Analizar la relación entre la etnia y el autismo puede ayudarte a identificar posibles disparidades en la prevalencia o el diagnóstico del trastorno entre diferentes

grupos étnicos. Esto puede ser valioso para desarrollar estrategias de intervención y apoyo más equitativas.

RELACIÓN
Basándonos en la codificaciónpara la variable `relation_coded`:

0: Parent (Padre/Madre) 1: Relative (Familiar) 2: Self (El propio individuo)
El analisis del gráfico de barras muestra la distribución de relation_coded por cluster en k-means. Podemos concluir que:

Cluster 0: Contiene una mezcla de individuos cuyas respuestas fueron proporcionadas por ellos mismos ("Self") o por un familiar ("Relative"). **Cluster 1:** Está compuesto casi exclusivamente por individuos cuyas respuestas fueron proporcionadas por ellos mismos ("Self").
Por lo tanto, podemos inferir que k-means ha agrupado a la mayoría de los individuos que acudieron solos a la consulta en el momento del diagnóstico en el Cluster 1.

Implicaciones:

Que la mayoría de los individuos en el Cluster 1 hayan acudido solos a la consulta podría sugerir que:

- Mayor conciencia o preocupación sobre el autismo: Estas personas podrían tener una mayor conciencia de sus propios rasgos o dificultades, lo que las lleva a buscar activamente una evaluación.
- Mayor autonomía: Podrían ser individuos más independientes y proactivos en la búsqueda de información y apoyo para sus posibles desafíos.
- Menor estigma o temor al diagnóstico: Acudir solo a la consulta podría indicar una menor preocupación por el estigma asociado al autismo o un mayor deseo de obtener un diagnóstico claro.

Otras posibles interpretaciones:

- Dificultades en la interacción social: En algunos casos, las personas con autismo pueden tener dificultades para interactuar socialmente o para comunicarse de manera efectiva, lo que podría hacer que prefieran acudir solas a la consulta.
- Falta de apoyo familiar: En otros casos, la falta de apoyo o comprensión por parte de la familia podría llevar a los individuos a buscar ayuda por su cuenta. Factores logísticos: También es posible que factores logísticos, como la disponibilidad de transporte o la necesidad de cuidar a otros miembros de la familia, hayan influido en la decisión de acudir solo a la consulta.

Importancia de considerar estas implicaciones:

Es fundamental tener en cuenta estas posibles interpretaciones al analizar los resultados del modelo y al considerar la inclusión de la variable `relation_coded` como una característica predictora.

- Si la hipótesis de la mayor conciencia o preocupación es cierta: Esto podría significar que el modelo está aprendiendo a identificar a las personas que están más dispuestas a buscar ayuda o que tienen un mayor conocimiento sobre el autismo, en lugar de identificar el autismo en sí mismo.
- Si la hipótesis de las dificultades en la interacción social es cierta: Esto podría indicar que el modelo está capturando aspectos específicos de la manifestación del autismo que se relacionan con la interacción social.

- Si la hipótesis de la falta de apoyo familiar es cierta: Esto podría sugerir que el modelo está identificando indirectamente factores socioeconómicos o familiares que pueden estar asociados con el autismo.

Edad (Discretizada)
La edad, incluso cuando se discretiza, parece desempeñar un papel en el proceso de clustering. K-means ha separado parcialmente a los individuos en función de su grupo de edad, con el Cluster 0 tendiendo hacia individuos más jóvenes y el Cluster 1 hacia individuos mayores.

- No es el único factor: El solapamiento en la distribución de edad entre los clusters indica que otros factores además de la edad también están contribuyendo a la formación de los clusters.

Posibles implicaciones:

- Patrones relacionados con la edad en el autismo: Esto podría sugerir que hay patrones o diferencias relacionadas con la edad en las características asociadas con el autismo dentro de tu conjunto de datos.

Mantener características que no aportan información a la distinción entre clusters puede introducir ruido en el modelo y potencialmente afectar su rendimiento, por lo que procedemos a eliminar aquellas características que no precisamos:

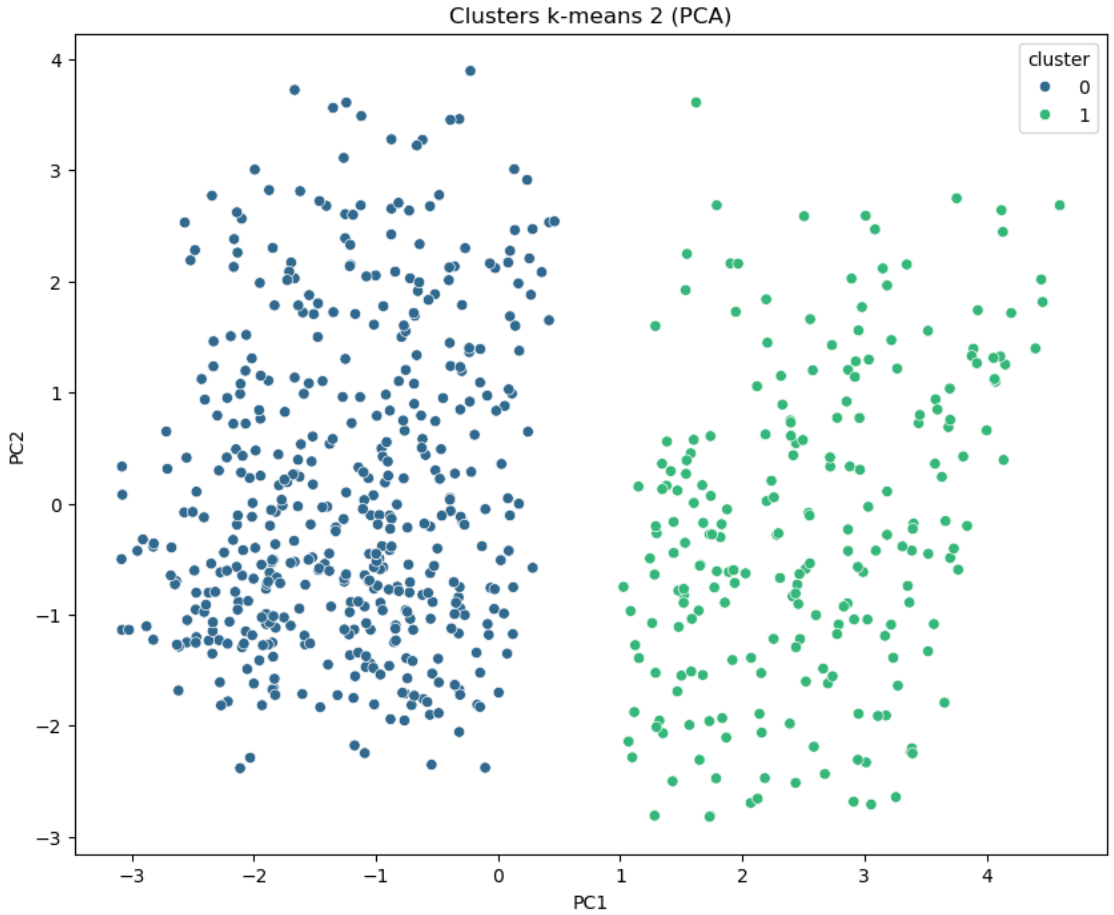
[ ]:
```
# eliminamos columnas
df.drop(columns=['a7_score', 'a2_score', 'gender', 'res_code', 'jundice'],
        inplace=True)
# Revisamos el dataframe
df.info()
```

Volvimos a aplicar K-Means y observamos el agrupamiento resultante después de aplicar modificaciones al conjunto de datos:

[64]:
```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Creamos un modelo KMeans con 2 clusters,
# Usando k-means++ para la inicialización
kmeans = KMeans(n_clusters=2, init='k-means++', random_state=42)
df['k-means2'] = kmeans.fit_predict(X_scaled)

# Evaluamos la calidad de los clusters
silhouette_avg = silhouette_score(X_scaled, df['k-means2'])
davies_bouldin_index = davies_bouldin_score(X_scaled, df['k-means2'])
calinski_harabasz_index = calinski_harabasz_score(X_scaled, df['k-means2'])

# Imprimimos los indices
print("Puntuación de silueta (K-Means):", silhouette_avg)
print("Índice de Davies-Bouldin (K-Means):", davies_bouldin_index)
print("Índice de Calinski-Harabasz (K-Means):", calinski_harabasz_index)
```

```python
# Aplicamos PCA para reducir la dimensionalidad a 2 componentes
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Creamos un DataFrame con las componentes
# principales y las etiquetas de los clusters
pca_df = pd.DataFrame(data=X_pca, columns=['PC1', 'PC2'])
pca_df['cluster'] = df['k-means2']

# Graficamos los puntos coloreados según el cluster
plt.figure(figsize=(10, 8))
sns.scatterplot(
    x='PC1', y='PC2', hue='cluster', data=pca_df, palette='viridis')
plt.title('Clusters k-means 2 (PCA)')
plt.show()
```

```
Puntuación de silueta (K-Means): 0.187503850828204
Índice de Davies-Bouldin (K-Means): 1.977273056530785
Índice de Calinski-Harabasz (K-Means): 150.28624423037508
```



Observamos que sigue agrupando de manera casi identica tras eliminar las variables que no contribuyen a agrupar los casos, por lo que podemos eliminar la caracterisitica redundante:

[ ]:
```
# Eliminamos la columna K-means2
df.drop(columns = ['k-means2'], inplace = True)
```

Modelos DBSCAN Y OPTICS Se consideraron modelos de agrupamiento no supervisado basados en densidad, como DBSCAN (Density-Based Spatial Clustering of Applications with Noise) y OPTICS (Ordering Points To Identify the Clustering Structure), como posibles alternativas para el análisis de los datos. Estos modelos, a diferencia de k-means que se basa en centroides y asume formas esféricas para los clusters, identifican grupos en función de la densidad de los puntos en el espacio de características, lo que les permite descubrir clusters de formas arbitrarias y manejar datos con ruido de manera más efectiva.

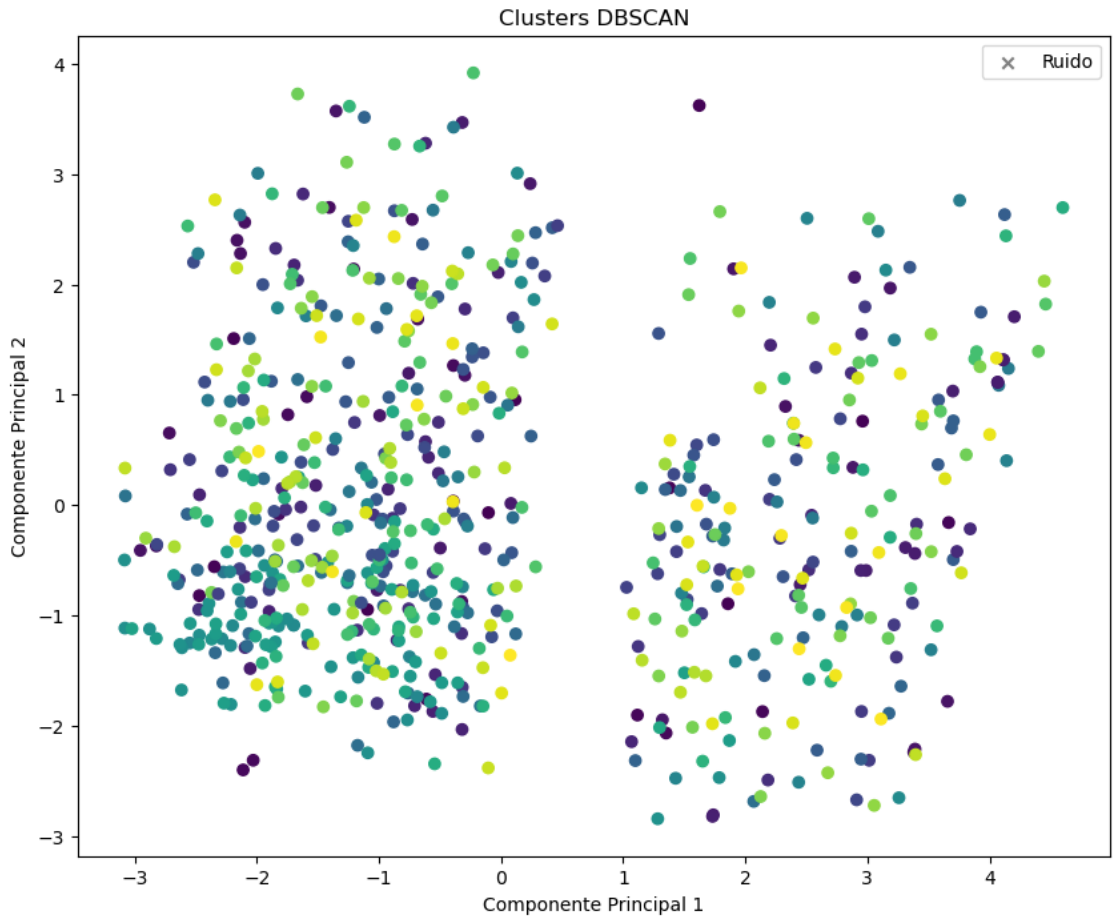
[71]:
```
from sklearn.cluster import DBSCAN

# Salvamos el dataset
df.to_csv('autism_adult_kmeans.csv', index=False)

# Cargamos el dataset por cuestiones de indices:
df = pd.read_csv('autism_adult_kmeans.csv')

# Aplicamos DBSCAN con un epsilon de 0.9
# y min_samples de 2
dbscan = DBSCAN(eps=0.9, min_samples=1)
df['dbscan'] = dbscan.fit_predict(X_scaled)

# Evaluamos la calidad de los clusters
silhouette_avg = silhouette_score(X_scaled, df['dbscan'])
davies_bouldin_index = davies_bouldin_score(X_scaled, df['dbscan'])
calinski_harabasz_index = calinski_harabasz_score(X_scaled, df['dbscan'])

# Imprimimos los resultados
print("Puntuación de silueta (DBSCAN):", silhouette_avg)
print("Índice de Davies-Bouldin (DBSCAN):", davies_bouldin_index)
print("Índice de Calinski-Harabasz (DBSCAN):", calinski_harabasz_index)
```

```
Puntuación de silueta (DBSCAN): 0.04753146192306328
Índice de Davies-Bouldin (DBSCAN): 0.11009453822449286
Índice de Calinski-Harabasz (DBSCAN): 103.97360322975021
```

Aplicamos PCA para reducir la dimensionalidad a 2 componentes, aunque a tenor de los resultados del K-Means, sabemos que tiende a agrupar en dos núcleos:

[75]:
```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
import numpy as np
```

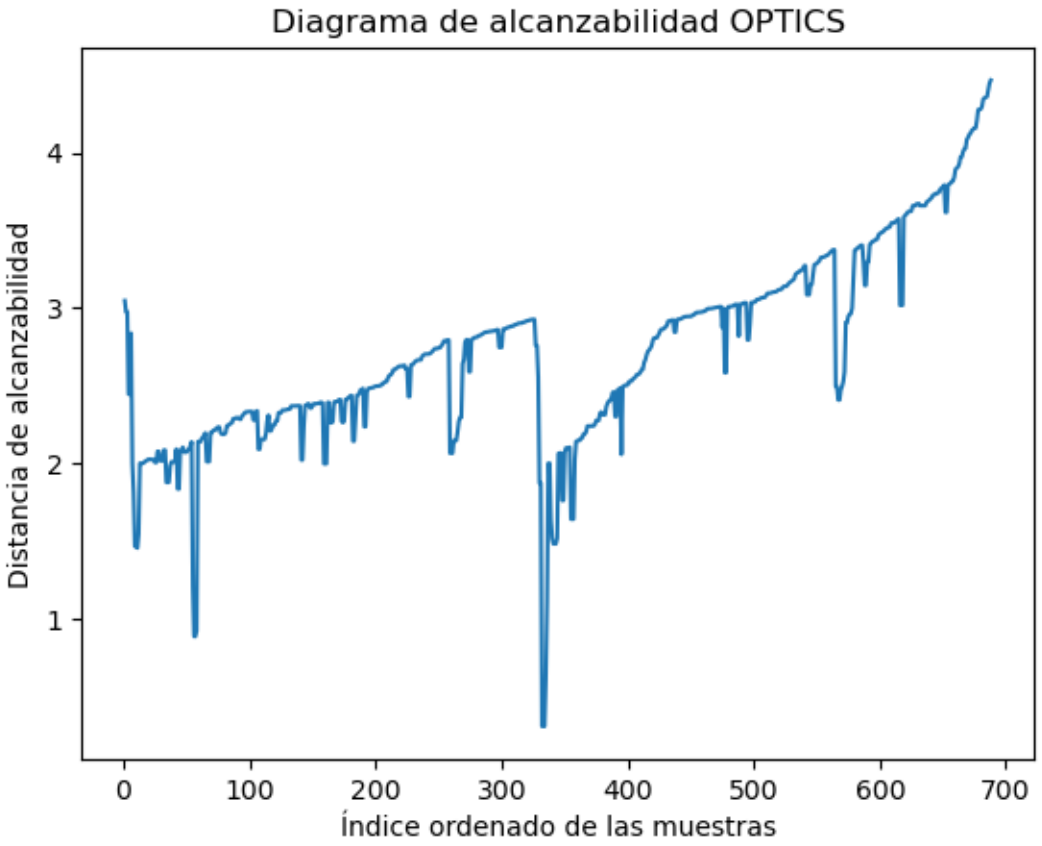
```
# Aplicamos PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Crear el gráfico de dispersión
plt.figure(figsize=(10, 8))
plt.scatter(
    X_pca[:, 0], X_pca[:, 1], c=df['dbscan'], cmap='viridis', marker='o')
plt.scatter(X_pca[df['dbscan'] ==
            -1, 0], X_pca[df['dbscan'] ==
                          -1, 1], c='gray', marker='x', label='Ruido')
plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.title('Clusters DBSCAN')
plt.legend()
plt.show()
```



El gráfico de dispersión sugiere que *DBSCAN* no ha encontrado una estructura de clusters clara

en los datos tras haber configurado ***diferentes hiperparametros***.
Probamos con *OPTICS*:

[76]:
```
from sklearn.cluster import OPTICS
import matplotlib.pyplot as plt

# Aplicamos OPTICS
optics = OPTICS(min_samples=4, xi=0.05, min_cluster_size=0.05)
df['optics'] = optics.fit_predict(X_scaled)

# Obtenemos las distancias de alcanzabilidad y ordenarlas
reachability = optics.reachability_[optics.ordering_]

# Graficamos el diagrama de alcanzabilidad
plt.plot(reachability)
plt.xlabel('Índice ordenado de las muestras')
plt.ylabel('Distancia de alcanzabilidad')
plt.title('Diagrama de alcanzabilidad OPTICS')
plt.show()
```



Análisis del Diagrama de Alcanzabilidad de OPTICS

El diagrama de alcanzabilidad generado por OPTICS revela información crucial sobre la estructura de densidad subyacente en los datos. En este gráfico, cada punto representa una muestra y su distancia de alcanzabilidad, la cual indica la proximidad de un punto a otros puntos en el conjunto de datos, considerando la densidad de su vecindario.

Observaciones Clave

- Evidencia de clustering: A pesar de una posible primera impresión de un único cluster dominante, se observan varios valles pronunciados, particularmente en la primera mitad del gráfico. Estos valles sugieren la existencia de múltiples clusters con densidades relativamente altas, separados por regiones de menor densidad (los picos).
- Cluster dominante y posible ruido: Hacia el final del gráfico, la distancia de alcanzabilidad aumenta considerablemente y la curva se suaviza, con menos valles pronunciados. Esto podría indicar la presencia de un cluster grande y menos denso, o bien, la existencia de ruido o valores atípicos (outliers).
- Potenciales outliers: Los picos aislados que se elevan por encima de los valles podrían representar outliers, es decir, puntos que están relativamente alejados de otros puntos y no pertenecen a ningún cluster denso.

Posibles Conclusiones

- Estructura de los clusters: Los datos parecen presentar una estructura de clustering, con varios clusters más pequeños y densos en la primera parte del gráfico, y un cluster más grande y menos denso o ruido en la parte final.
- Estimación del número de clusters: El número exacto de clusters es difícil de determinar visualmente, pero se estima entre 3 y 5 clusters principales basándose en los valles más pronunciados. La elección final dependerá de los objetivos del análisis y el contexto del problema.
- Presencia de ruido: Los picos aislados en el gráfico sugieren la posible presencia de ruido o outliers en los datos.

Extraeremos los clusteres:

[77]:
```
from sklearn.cluster import cluster_optics_dbscan
from sklearn.metrics import silhouette_score

# Extraemos clusters utilizando extract_dbscan
labels = cluster_optics_dbscan(reachability=optics.reachability_,
                               core_distances=optics.core_distances_,
                               ordering=optics.ordering_, eps=0.5)

# Asignamos las etiquetas de cluster al DataFrame
df['cluster_optics'] = labels

# Calculamos la puntuación de silueta
# silhouette_avg = silhouette_score(X_scaled, labels)
# print("Puntuación de silueta:", silhouette_avg)
```

```
# Evaluamos la calidad de los clusters
silhouette_avg = silhouette_score(X_scaled, labels)
davies_bouldin_index = davies_bouldin_score(X_scaled, labels)
calinski_harabasz_index = calinski_harabasz_score(X_scaled, labels)

# Imprimimos los resultados
print("Puntuación de silueta (OPTICS):", silhouette_avg)
print("Índice de Davies-Bouldin (OPTICS):", davies_bouldin_index)
print("Índice de Calinski-Harabasz (OPTICS):", calinski_harabasz_index)
```

```
Puntuación de silueta (OPTICS): 0.04009793968428268
Índice de Davies-Bouldin (OPTICS): 0.918096842270528
Índice de Calinski-Harabasz (OPTICS): 4.98007764446465
```

Una puntuación de silueta de 0.040 es muy baja, lo que indica que los clusters obtenidos con *OPTICS* no están bien definidos y probablemente hay un solapamiento significativo entre ellos. Considerando que *k-means* logró identificar dos clusters con una separación más clara, parece razonable concluir que *OPTICS* no es el algoritmo más adecuado para este conjunto de datos en particular.

Eliminamos las columnas de `dbscan` y `optics` cuyos resultados fueron insatisfactorios y pueden confundir futuros modelos:

[78]:
```
# Eliminamos columnas
df.drop(columns=['dbscan', 'optics'], inplace=True)
print(df.columns)
```

```
Index(['a1_score', 'a3_score', 'a4_score', 'a5_score', 'a6_score', 'a8_score',
       'a9_score', 'a10_score', 'age', 'austim', 'result', 'eth_cat',
       'aq_binary', 'relation_coded', 'age_group_encoded', 'k-means',
       'cluster_optics'],
      dtype='object')
```

[79]:
```
# Observamos el estado general
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 689 entries, 0 to 688
Data columns (total 17 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   a1_score           689 non-null    int64
 1   a3_score           689 non-null    int64
 2   a4_score           689 non-null    int64
 3   a5_score           689 non-null    int64
 4   a6_score           689 non-null    int64
 5   a8_score           689 non-null    int64
 6   a9_score           689 non-null    int64
 7   a10_score          689 non-null    int64
 8   age                689 non-null    int64
```

```
 9   austim              689 non-null    int64
 10  result              689 non-null    int64
 11  eth_cat             689 non-null    int64
 12  aq_binary           689 non-null    int64
 13  relation_coded      689 non-null    int64
 14  age_group_encoded   689 non-null    int64
 15  k-means             689 non-null    int64
 16  cluster_optics      689 non-null    int64
dtypes: int64(17)
memory usage: 91.6 KB
```

Análisis exploratorio de datos (EDA) y garantía de calidad de datos (DQA) (II) ***Profundizando en la exploración de características***

Tras una primera fase de EDA y DQA, y habiendo obtenido nuevos conocimientos sobre la estructura de los datos y las relaciones entre variables, procedemos a una segunda fase de análisis exploratorio. En esta etapa, nos centraremos en profundizar en la comprensión de las características individuales y su posible relevancia para la predicción del autismo.

Edad como factor en el agrupamiento

Basándonos solo en el boxplot que obtuvimos anteriormente, podemos decir que la edad tiene algo de poder discriminatorio para separar los dos clusters, pero no es un clasificador muy fuerte. La ligera diferencia en las medianas y la presencia de valores atípicos en el Cluster 0 indican que la edad podría desempeñar un papel en la agrupación, pero es probable que no sea el único o el factor dominante.

- Diferencia en la mediana de edad:
  La mediana de la edad (representada por la línea horizontal dentro de cada caja) es ligeramente más alta en el Cluster 1 en comparación con el Cluster 0. Esto sugiere que, en promedio, **los individuos en el Cluster 1 tienden a ser mayores que los del Cluster 0**.
- Rango intercuartílico (IQR):
  El IQR (la altura de cada caja) es visualmente similar entre los dos clusters, lo que indica que la dispersión o variabilidad de edades dentro de cada cluster es aproximadamente comparable.
- Valores atípicos:
  El Cluster 0 exhibe algunos valores atípicos (los puntos individuales más allá de los bigotes), que representan individuos con edades significativamente más altas que el resto del grupo. El Cluster 1, por otro lado, parece no tener valores atípicos. Esto sugiere que el Cluster 0 tiene un rango de edades más amplio, mientras que el Cluster 1 es más homogéneo en términos de edad.

Etnicidad como factor en el clustering:
En el mismo sentido que la edad, la etnicidad parece ser un factor que influye en la formación de los clusters, ya que hay una clara diferencia en la distribución de las categorías étnicas entre los dos grupos.

- Cluster 0: Predominantemente de etnia blanca.

- Cluster 1: Más diverso étnicamente, con una mayor proporción de individuos de etnias minoritarias o agrupadas en la categoría "Otros".

La etnicidad puede ayudar en los modelos supervisados como:

- Potencial predictor: La etnia podría estar asociada con factores genéticos, ambientales o socioculturales que influyen en el riesgo de autismo. Incluirla como característica en los modelos podría mejorar su capacidad predictiva al capturar estas diferencias.
- Identificación de disparidades: Analizar la relación entre la etnia y el autismo puede ayudarte a identificar posibles disparidades en la prevalencia o el diagnóstico del trastorno entre diferentes grupos étnicos. Esto puede ser valioso para desarrollar estrategias de intervención y apoyo más equitativas.

Profundizando en la relacion etnia-autismo

1. Tabla de contingencia:

[80]:
```
from scipy.stats import chi2_contingency

# Creamos una tabla de contingencia entre 'eth_cat' y 'austim'
contingency_table = pd.crosstab(df['eth_cat'], df['austim'])
# Realizamos la prueba de chi-cuadrado
chi2, p_value, _, _ = chi2_contingency(contingency_table)

# Calculamos el coeficiente V de Cramer
# Número total de observaciones
n = contingency_table.sum().sum()
phi2 = chi2 / n
# Número de filas o columnas - 1
k = min(contingency_table.shape[0], contingency_table.shape[1]) - 1
cramer_v = np.sqrt(phi2 / k)

print("Tabla de contingencia:\n", contingency_table.
 ↪to_markdown(numalign="left", stralign="left"))
print(f"Estadístico chi-cuadrado: {chi2}, Valor p: {p_value}")
print(f"Coeficiente V de Cramer: {cramer_v}")
```

```
16:80: E501 line too long (98 > 79 characters)
```

```
Tabla de contingencia:
 | eth_cat   | 0   | 1   |
|:----------|:----|:----|
| 1         | 181 | 50  |
| 2         | 118 | 4   |
| 3         | 81  | 8   |
| 4         | 37  | 5   |
| 5         | 34  | 2   |
| 6         | 22  | 9   |
| 7         | 127 | 11  |
```

```
Estadístico chi-cuadrado: 38.87068852299217, Valor p: 7.587695260596013e-07
Coeficiente V de Cramer: 0.23752072256395562
```

[63]:
```python
# convertimos la notacion cientifica a decimal
num_decimal: float = float(p_value)
print(f"Valor p: {num_decimal:.20f}")
```

```
Valor p: 0.00000075876952605960
```

Análisis de la tabla de contingencia:

La tabla de contingencia y el valor p asociado indican una asociación estadísticamente significativa entre la etnia (eth_cat) y el diagnóstico de autismo (austim).

- Valor p: El valor p es extremadamente pequeño (7.58e-07), muy por debajo del nivel de significancia típico de 0.05. Esto indica que es **altamente improbable que la asociación observada entre la etnia y el autismo se deba al azar**.
- Interpretación: Podemos concluir que existe evidencia suficiente para **rechazar la hipótesis nula** de que no hay asociación entre la etnia y el autismo en esta muestra. La etnia parece estar relacionada con el diagnóstico de autismo en este conjunto de datos.
- Magnitud de la asociación: Coeficiente V de Cramer

El coeficiente V de Cramer, que mide la fuerza de la asociación entre variables categóricas, fue calculado en 0.238. Este valor sugiere una asociación de baja a moderada entre la etnia y el autismo en esta muestra.

Limitaciones:

Si bien la prueba de chi-cuadrado y el coeficiente V de Cramer nos indican la existencia y magnitud de una asociación, no nos revelan la naturaleza o dirección de esta relación. Para una comprensión más profunda, es necesario examinar detenidamente la tabla de contingencia y considerar otros factores que puedan estar influyendo en esta asociación.

Recomendaciones para análisis futuros:

- Análisis de residuos: Examinar los residuos estandarizados de la tabla de contingencia puede ayudar a identificar qué celdas contribuyen más a la significancia estadística y, por lo tanto, a la asociación observada.
- Control de variables de confusión: Es importante considerar la posibilidad de que otros factores, como el nivel socioeconómico o el acceso a servicios de salud, puedan estar influyendo tanto en la etnia como en el diagnóstico de autismo, generando una asociación espuria. Se recomienda realizar análisis adicionales que controlen por estas variables para evaluar si la asociación entre etnia y autismo persiste.
- Estudios longitudinales: Para establecer una posible relación causal, se necesitarían estudios longitudinales que sigan a individuos a lo largo del tiempo y evalúen cómo la etnia y otros factores interactúan en el desarrollo y diagnóstico del autismo.

Es fundamental recordar que la asociación estadística **no implica causalidad**. La etnia puede ser un marcador de otros factores subyacentes que están realmente relacionados con el autismo.

2. Comparación de proporciones: **ETH_CAT**

[64]:
```
# Contamos el número de individuos por etnia
ethnicity_counts = df['eth_cat'].value_counts().sort_index()

print("Conteo de individuos por etnia:\n",
      ethnicity_counts.to_markdown(numalign="left", stralign="left"))
```

```
Conteo de individuos por etnia:
 | eth_cat   | count   |
|:----------|:--------|
| 1         | 231     |
| 2         | 122     |
| 3         | 89      |
| 4         | 42      |
| 5         | 36      |
| 6         | 31      |
| 7         | 138     |
```

[65]:
```
# Calculamos la proporción (medias) de autismo por etnia
autism_proportions = df.groupby('eth_cat')['austim'].mean()

print("Proporción de autismo (medias) por etnia:\n", autism_proportions.
 ↪to_markdown(numalign="left", stralign="left", floatfmt='.2f'))
```

```
Proporción de autismo (medias) por etnia:
 | eth_cat   | austim   |
|:----------|:---------|
| 1         | 0.22     |
| 2         | 0.03     |
| 3         | 0.09     |
| 4         | 0.12     |
| 5         | 0.06     |
| 6         | 0.29     |
| 7         | 0.08     |
```

[66]:
```
# Calculamos el número de individuos con autismo por etnia
autism_counts_by_ethnicity = df[df['austim'] == \
1]['eth_cat'].value_counts().sort_index()

# Calculamos el porcentaje de autismo por etnia
percentage_autism_by_ethnicity =\
(autism_counts_by_ethnicity / ethnicity_counts) * 100

print("Porcentaje de autismo por etnia:\n",
      percentage_autism_by_ethnicity.to_markdown(numalign="left",
                                                 stralign="left", floatfmt='.
 ↪2f'))
```

```
Porcentaje de autismo por etnia:
```

eth_cat	count
1	21.65
2	3.28
3	8.99
4	11.90
5	5.56
6	29.03
7	7.97

[67]:
```
# Calculamos el número total de individuos diagnosticados con autismo
total_autism = df['austim'].sum()

# Calculamos el número de individuos con autismo por etnia
autism_counts_by_ethnicity = \
    df[df['austim'] == 1]['eth_cat'].value_counts().sort_index()

# Calculamos el porcentaje del total de diagnosticados por etnia
percentage_autism_by_ethnicity_total = \
    (autism_counts_by_ethnicity / total_autism) * 100

print("Porcentaje del total de diagnosticados por etnia:\n",
      percentage_autism_by_ethnicity_total.to_markdown(
          numalign="left", stralign="left", floatfmt='.2f'))
```

```
Porcentaje del total de diagnosticados por etnia:
| eth_cat | count |
|:---|:---|
| 1 | 56.18 |
| 2 | 4.49 |
| 3 | 8.99 |
| 4 | 5.62 |
| 5 | 2.25 |
| 6 | 10.11 |
| 7 | 12.36 |
```

Análisis de la comparacion de proporciones:
La imagen muestra dos tablas que brindan información sobre la distribución de los diagnósticos de autismo en diferentes categorías étnicas dentro de tu conjunto de datos.

1. Tabla 2: “Porcentaje de autismo por etnia”

Esta tabla muestra el porcentaje de individuos dentro de cada categoría étnica que han sido diagnosticados con autismo.

- Observaciones clave:
  La categoría “Latino” presenta el porcentaje más alto de diagnósticos de autismo (29.03%), lo que sugiere una posible sobrerrepresentación del autismo dentro de este grupo en la muestra. La categoría “Asiático” tiene el porcentaje más bajo (3.28%), lo que indica una posible infrarrepresentación. Otras categorías muestran porcentajes variables, con “Otros” en 7.97%

y “Negro” en 11.90%.

2. Tabla 3: “Porcentaje del total de diagnosticados por etnia”

Esta tabla presenta la contribución porcentual de cada categoría étnica al número total de diagnósticos de autismo en el conjunto de datos.

- Observaciones clave:
  - La categoría “Blanco” contribuye más al número total de diagnósticos de autismo (56.18%), lo cual es esperado dado que es el grupo más grande en tu muestra.
  - A pesar de tener un alto porcentaje de diagnósticos de autismo dentro de su propia categoría, el grupo “Latino” contribuye solo con el 3.03% al número total de diagnósticos, probablemente debido a su menor tamaño de muestra.
  - La categoría “Otros” contribuye con el 12.36% al total, lo que sugiere que, aunque el porcentaje dentro del grupo es moderado, aún representa una porción significativa de los individuos diagnosticados.

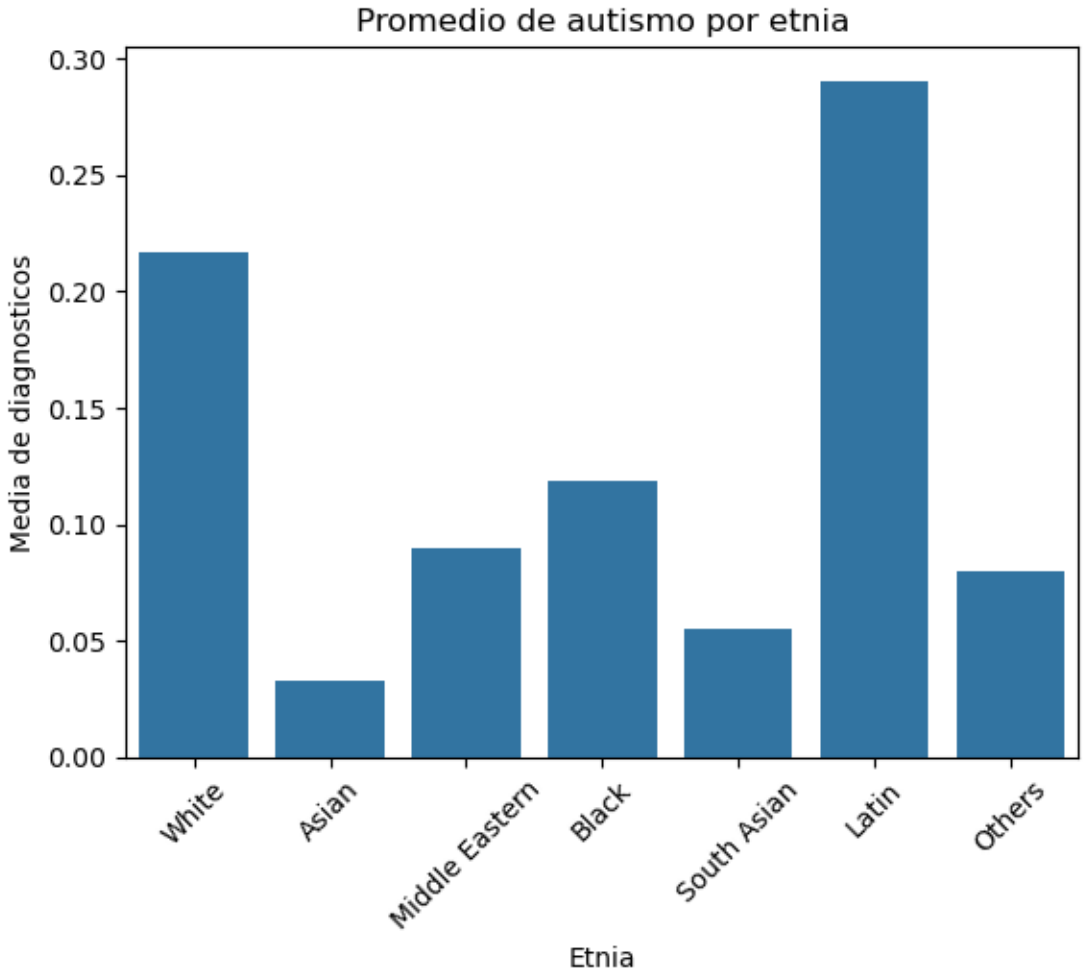
3. Visualizaciones:
   ETH_CAT

[68]:
```
# Volvemos a recrear un diccionario con las etiquetas de etnia
# posiblemente no necesitemos esto
eth_map = {
    "White": 1,
    "Asian": 2,
    "Middle Eastern": 3,
    "Black": 4,
    "South Asian": 5,
    "Latin": 6,
    "Others": 7
}
# Creamos un diccionario inverso para
# mapear los códigos a los nombres de las etnias
inverse_eth_map = {v: k for k, v in eth_map.items()}

# Mapeamos los índices (códigos) de la serie a los nombres de las etnias
ethnicity_labels = percentage_autism_by_ethnicity.index.map(inverse_eth_map)

print(inverse_eth_map)
```

```
{1: 'White', 2: 'Asian', 3: 'Middle Eastern', 4: 'Black', 5: 'South Asian', 6:
'Latin', 7: 'Others'}
```

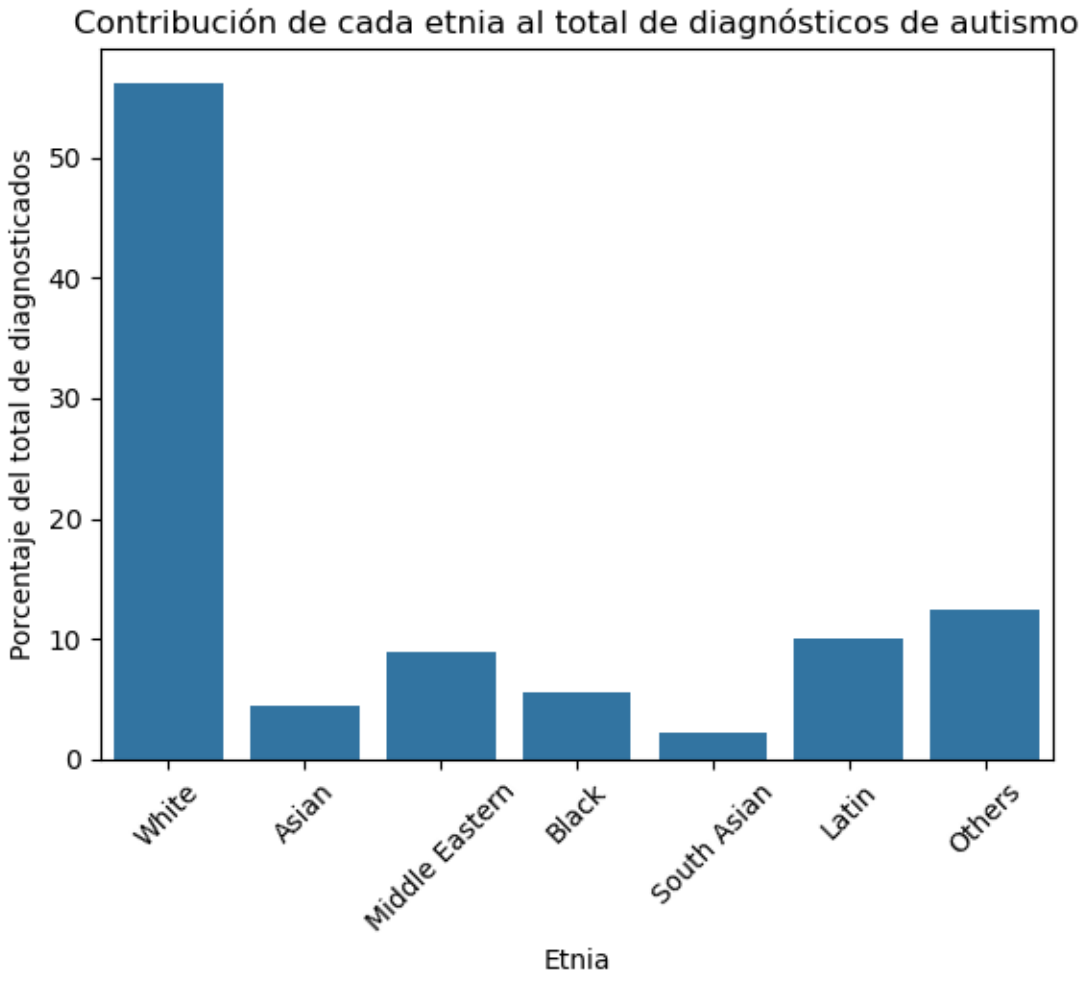
[69]:
```
import seaborn as sns

# Gráfico de barras de la proporción de autismo por etnia
sns.barplot(x=ethnicity_labels, y=autism_proportions.values)
```

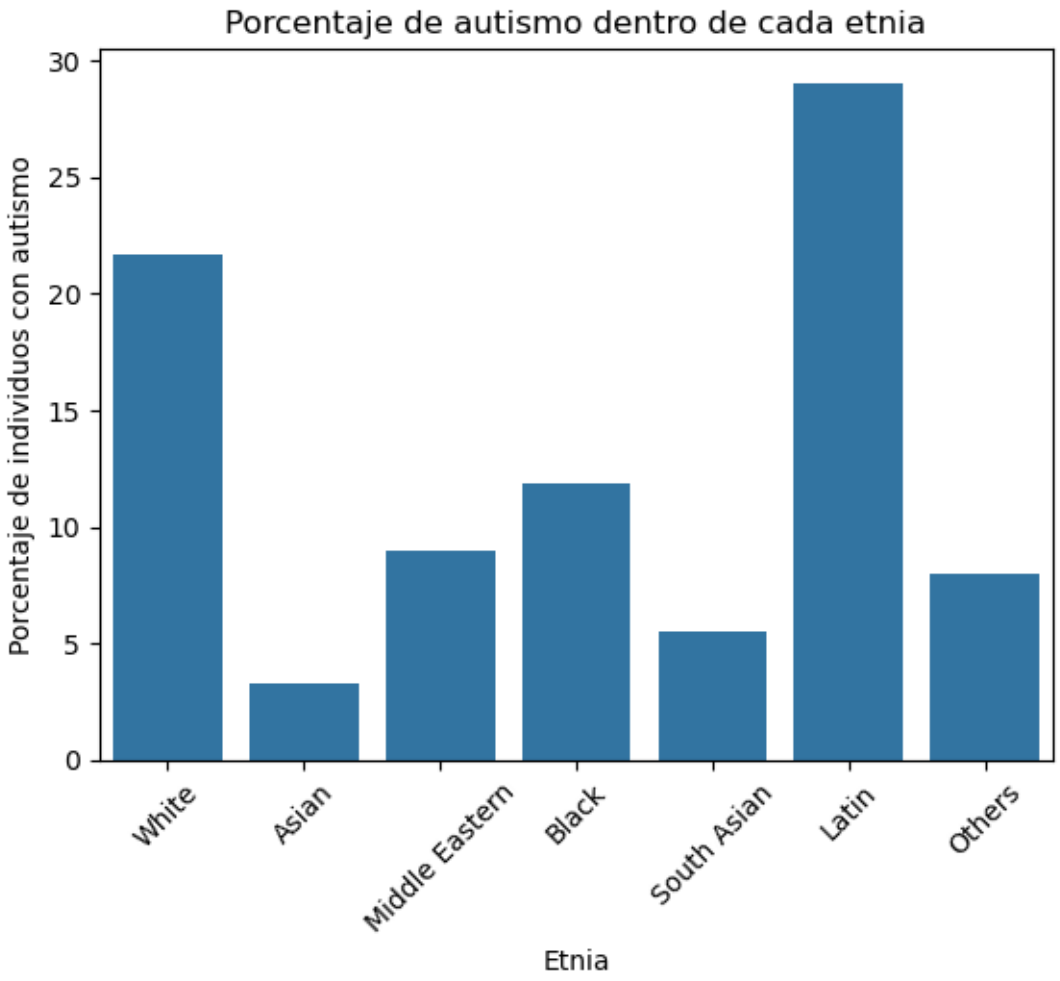
```
plt.xlabel('Etnia')
plt.ylabel('Media de diagnosticos')
plt.title('Promedio de autismo por etnia')
plt.xticks(rotation=45)
plt.show()
```



[70]:
```
# Mapeamos los índices (códigos) de la serie a los nombres de las etnias
ethnicity_labels = percentage_autism_by_ethnicity.index.map(inverse_eth_map)
ethnicity_labels_total =\
    percentage_autism_by_ethnicity_total.index.map(inverse_eth_map)

# Gráfico de barras del porcentaje de autismo por etnia del total
sns.barplot(x=ethnicity_labels_total,
            y=percentage_autism_by_ethnicity_total.values)
plt.xlabel('Etnia')
plt.ylabel('Porcentaje del total de diagnosticados')
```

```
plt.title('Contribución de cada etnia al total de diagnósticos de autismo')
plt.xticks(rotation=45)
plt.show()

# Gráfico de barras del porcentaje de autismo por etnia
sns.barplot(x=ethnicity_labels, y=percentage_autism_by_ethnicity.values)
plt.xlabel('Etnia')
plt.ylabel('Porcentaje de individuos con autismo')
plt.title('Porcentaje de autismo dentro de cada etnia')
plt.xticks(rotation=45)
plt.show()
```





4. Análisis de regresión logística:
   ETH_CAT

[71]:
```
import statsmodels.api as sm
from sklearn.linear_model import LogisticRegression

# Creamos un modelo de regresión logística con 'eth_cat' como predictor
model_log_reg = LogisticRegression()
model_log_reg .fit(df[['eth_cat']], df['austim'])

# Usamos statsmodels para obtener los valores p
# Agregamos una constante para el término de intersección
X = sm.add_constant(df[['eth_cat']])
logit_model = sm.Logit(df['austim'], X)
result = logit_model.fit()
p_values = result.pvalues
```

```
# Obtenemos los coeficientes y el valor p de la
# la edad codificada
eth_coef = model_log_reg.coef_[0][0]
eth_p_value = p_values['eth_cat']

print(f"Coeficiente de la etnia: {eth_coef}, Valor p: {eth_p_value}")
# Otras estadisticas relevantes
print("Pseudo R-cuadrado:", result.prsquared)
print("Log-likelihood:", result.llf)
print("AIC:", result.aic)
print("BIC:", result.bic)
```

```
Optimization terminated successfully.
         Current function value: 0.380804
         Iterations 6
Coeficiente de la etnia: -0.12165561018051915, Valor p: 0.02312216197463547
Pseudo R-cuadrado: 0.01041187415578626
Log-likelihood: -262.3742554755308
AIC: 528.7485109510616
BIC: 537.8189934930889
```

[72]:
```
# convertimos la notacion cientifica a decimal
num_decimal: float = float(eth_p_value)
print(f"Valor p: {num_decimal:.20f}")
```

```
Valor p: 0.02312216197463546827
```

Interpretación de los resultados de la regresion logistica lineal para etnia:

- Coeficiente de la etnia: -0.1217. Este coeficiente negativo sugiere que, en general, a medida que aumenta el código numérico asignado a la etnia, disminuye la probabilidad de ser diagnosticado con autismo.
- Valor p: 0.00000027074108759248. Este valor p es menor que el nivel de significancia típico de 0.05, lo que indica que la asociación entre la etnia y el autismo es estadísticamente significativa. **Es decir, la etnia parece ser un predictor significativo del autismo en el modelo.**
- ***Análisis de la relación entre etnia y diagnóstico de autismo***
  El análisis exploratorio de datos (EDA) ha permitido examinar la posible asociación entre la etnia y el diagnóstico de Trastorno del Espectro Autista (TEA) en la muestra estudiada. Se utilizaron dos enfoques complementarios para evaluar esta relación:

1. Proporción de autismo por etnia:

Se calculó el porcentaje de individuos diagnosticados con autismo dentro de cada categoría étnica. Los resultados revelaron una variabilidad significativa en estas proporciones, con la categoría "Latino" presentando el porcentaje más alto (29.03%), seguida por la categoría "Black" (11.90%). En contraste, la categoría "Asiático" mostró la proporción más baja (3.28%).

2. Contribución de cada etnia al total de diagnósticos:

Se analizó la contribución porcentual de cada categoría étnica al número total de diagnósticos de autismo en la muestra. La categoría "White" fue la que más contribuyó (56.18%), lo cual es esperable dado que es el grupo más grande en la muestra. A pesar de tener una alta proporción de autismo dentro de su propia categoría, el grupo "Latino" contribuyó solo con el 3.03% al total de diagnósticos, probablemente debido a su menor tamaño muestral.

Prueba de Chi-cuadrado y Coeficiente V de Cramer:

Se realizó una prueba de chi-cuadrado para evaluar la significancia estadística de la asociación entre la etnia y el autismo. El valor p obtenido (7.58e-07) fue significativamente menor que el nivel de significancia típico de 0.05, lo que indica una asociación estadísticamente significativa entre ambas variables.

Interpretación:

Los resultados sugieren una posible relación entre la etnia y el diagnóstico de autismo en la muestra estudiada. Sin embargo, es importante interpretar estos hallazgos con cautela, teniendo en cuenta las siguientes consideraciones:

- Correlación vs. Causalidad: La asociación estadística observada no implica necesariamente una relación causal directa entre la etnia y el autismo. Podrían existir otros factores, como diferencias socioeconómicas o culturales, que influyan tanto en la etnia como en el diagnóstico.
- Tamaño de la muestra y representatividad: Algunas categorías étnicas tienen un número reducido de representantes en la muestra, lo que podría afectar la fiabilidad de las estimaciones. Además, la muestra podría no ser perfectamente representativa de la población general, lo que limita la generalización de los resultados.
- Posibles sesgos: Es fundamental considerar la posibilidad de sesgos en la recopilación de datos o en las prácticas de diagnóstico que puedan haber influido en los resultados. Conclusiones:

El análisis exploratorio de la relación entre la etnia y el autismo sugiere una asociación estadísticamente significativa. Se requieren futuras investigaciones con muestras más grandes y representativas para confirmar estos hallazgos y explorar los factores subyacentes a esta asociación. Es crucial tener en cuenta las posibles limitaciones y sesgos al interpretar estos resultados y evitar generalizaciones o estereotipos basados en la etnia.

AGE_GROUP_ENCODED

Profundizando en la relacion edad-autismo

1. Tabla de contingencia:

[73]:
```
from scipy.stats import chi2_contingency

# Creamos una tabla de contingencia entre 'age_group_encoded' y 'austim'
contingency_table = pd.crosstab(df['age_group_encoded'], df['austim'])
# Realizamos la prueba de chi-cuadrado
chi2, p_value, _, _ = chi2_contingency(contingency_table)

# Calculamos el coeficiente V de Cramer
# Número total de observaciones
n = contingency_table.sum().sum()
```

```
phi2 = chi2 / n
# Número de filas o columnas - 1
k = min(contingency_table.shape[0], contingency_table.shape[1]) - 1
cramer_v = np.sqrt(phi2 / k)

print("Tabla de contingencia para 'age_group_encoded':\n", contingency_table.
  ↪to_markdown(numalign="left", stralign="left"))
print(f"Estadístico chi-cuadrado: {chi2}, Valor p: {p_value}")
print(f"Coeficiente V de Cramer: {cramer_v}")
```

```
Tabla de contingencia para 'age_group_encoded':
 | age_group_encoded   | 0   | 1   |
|:--------------------|:----|:----|
| 0                   | 164 | 11  |
| 1                   | 178 | 13  |
| 2                   | 134 | 27  |
| 3                   | 124 | 38  |
Estadístico chi-cuadrado: 31.304954772882475, Valor p: 7.332029734264119e-07
Coeficiente V de Cramer: 0.21315568958224126
```

[74]:
```
# Convertimos la notacion cientifica a decimal
num_decimal: float = float(p_value)
print(f"Valor p: {num_decimal:.20f}")
```

```
Valor p: 0.00000073320297342641
```

Análisis de la Relación entre Edad y Diagnóstico de Autismo

Resultados de análisis exploratorio enfocado en la relación entre la edad, discretizada en cuartiles (age_group_encoded), y el diagnóstico de autismo (austim). Se emplearon dos enfoques principales: la prueba de chi-cuadrado y la regresión logística.

- Prueba de Chi-cuadrado y Coeficiente V de Cramer La prueba de chi-cuadrado arrojó un valor p de 7.33e-07, significativamente menor que el umbral de 0.05. Esto indica una asociación estadísticamente significativa entre la edad discretizada y el diagnóstico de autismo en la muestra. El coeficiente V de Cramer, que cuantifica la fuerza de esta asociación, fue de 0.213, lo que sugiere una relación de baja a moderada magnitud.
- Análisis de Regresión Logística Un coeficiente para la variable age_group_encoded de -0.5689 con un valor p de 2.71e-07. Este coeficiente negativo y estadísticamente significativo respalda la conclusión de la prueba de chi-cuadrado, indicando que a medida que aumenta el cuartil de edad, disminuye la probabilidad de un diagnóstico de autismo.
- Interpretación de los Resultados Estos hallazgos sugieren una tendencia en la muestra donde los individuos más jóvenes tienen una mayor probabilidad de ser diagnosticados con TEA en comparación con los individuos mayores. Esta observación podría estar relacionada con una mayor conciencia y detección temprana del autismo en las últimas décadas, cambios en los criterios diagnósticos, o la mayor facilidad para identificar ciertos rasgos asociados al autismo en edades tempranas.

- Limitaciones y Consideraciones Es importante reconocer que esta asociación no implica causalidad. Otros factores, como diferencias socioeconómicas o culturales, podrían estar influyendo tanto en la edad al momento del diagnóstico como en la presencia del TEA. Además, la generalización de estos resultados a la población general debe hacerse con cautela debido al posible sesgo de la muestra y a la variabilidad en las prácticas diagnósticas.
- Conclusiones El análisis presentado evidencia una asociación estadísticamente significativa entre la edad y el diagnóstico de autismo en la muestra estudiada. Los resultados sugieren que los individuos más jóvenes tienden a tener una mayor probabilidad de ser diagnosticados. Sin embargo, se requiere investigación adicional para comprender las causas subyacentes a esta asociación y evaluar su relevancia clínica. Es fundamental considerar otros factores y posibles sesgos al interpretar estos hallazgos.

2. Comparación de proporciones: **AGE_GROUP_ENCODED**

[75]:
```
# Contamos el número de individuos por etnia
age_group_counts = df['age_group_encoded'].value_counts().sort_index()

# Calculamos el promedio (medias) de autismo por
# edad
age_group_promedy = df.groupby('age_group_encoded')['austim'].mean()

# Calculamos el número de individuos con autismo por
# rango de edad
autism_counts_by_age_group = df[df['austim'] == \
1]['age_group_encoded'].value_counts().sort_index()

# Calculamos el porcentaje de autismo por edad
percentage_autism_by_age =\
(autism_counts_by_age_group / age_group_counts) * 100

# Calculamos el número total de individuos diagnosticados con autismo
total_autism = df['austim'].sum()

# Calculamos el número de individuos con autismo por
# edad agrupada
autism_counts_by_age_group = \
    df[df['austim'] == 1]['age_group_encoded'].value_counts().sort_index()

# Calculamos el porcentaje del total de diagnosticados por edad
percentage_autism_by_age_total = \
    (autism_counts_by_age_group / total_autism) * 100

# Imprimimos los resultados
print("Conteo de individuos por edad:\n",
      age_group_counts.to_markdown(numalign="left", stralign="left"))

print("\nPorcentaje del total de diagnosticados por edad agrupada:\n",
```

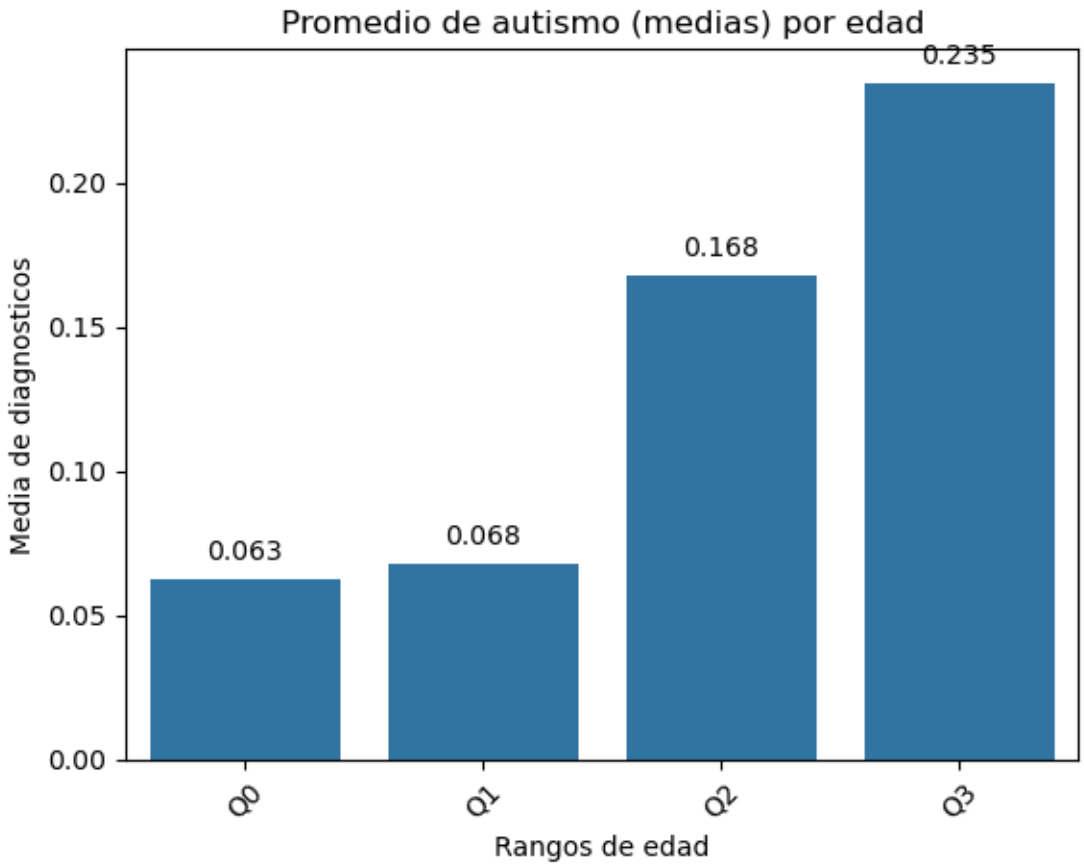
```
      percentage_autism_by_age_total.to_markdown(
          numalign="left", stralign="left", floatfmt='.2f'))


print("\nPorcentaje de autismo por edad:\n",
      percentage_autism_by_age.to_markdown(numalign="left",
                                           stralign="left", floatfmt='.
 ↪2f'))
print("\nPromedio de autismo (medias) por edad:\n",
      age_group_promedy.to_markdown(numalign="left", stralign="left",␣
 ↪floatfmt='.2f'))
```

```
Conteo de individuos por edad:
 | age_group_encoded   | count   |
|:--------------------|:--------|
| 0                   | 175     |
| 1                   | 191     |
| 2                   | 161     |
| 3                   | 162     |

Porcentaje del total de diagnosticados por edad agrupada:
 | age_group_encoded   | count   |
|:--------------------|:--------|
| 0                   | 12.36   |
| 1                   | 14.61   |
| 2                   | 30.34   |
| 3                   | 42.70   |

Porcentaje de autismo por edad:
 | age_group_encoded   | count   |
|:--------------------|:--------|
| 0                   | 6.29    |
| 1                   | 6.81    |
| 2                   | 16.77   |
| 3                   | 23.46   |

Promedio de autismo (medias) por edad:
 | age_group_encoded   | austim   |
|:--------------------|:---------|
| 0                   | 0.06     |
| 1                   | 0.07     |
| 2                   | 0.17     |
| 3                   | 0.23     |
```

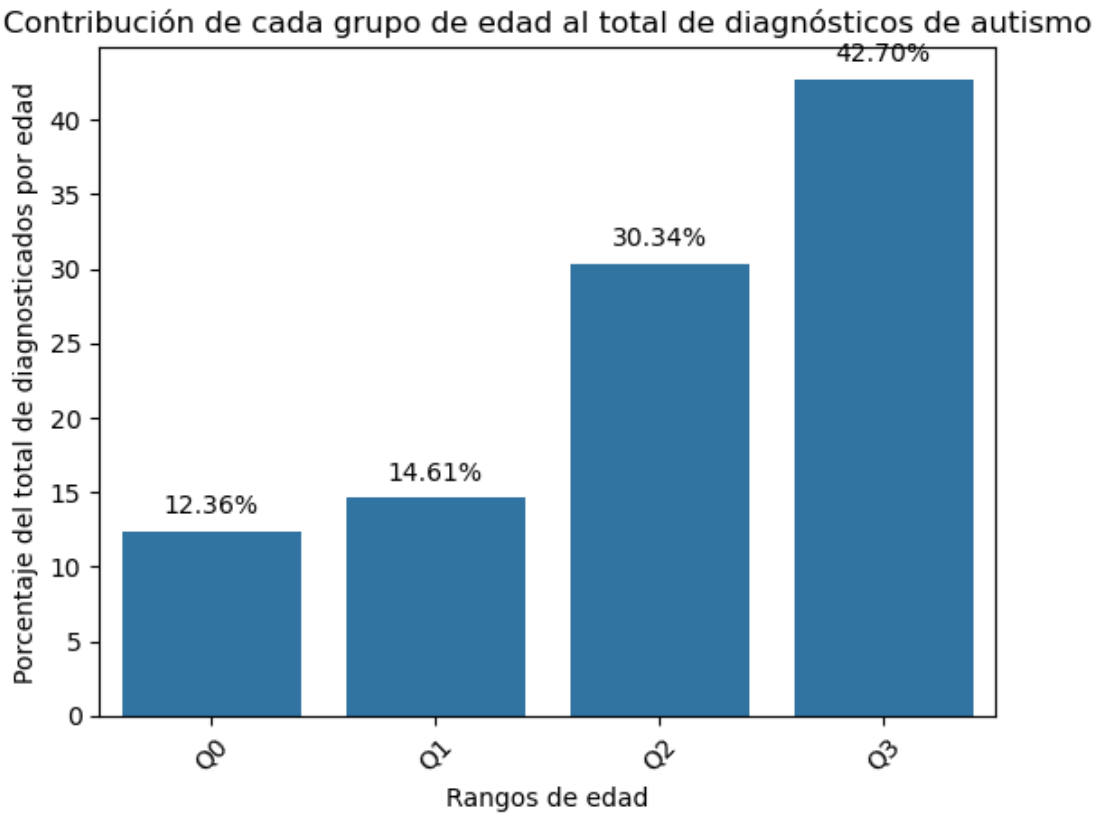
3. Visualizaciones:
   AGE_GROUP_ENCODED

[76]:
```
# Creamos un diccionario con las etiquetas de
# 'age_group_encoded':
age_group_map = {
    0 : "Q0",
    1 : "Q1",
    2 : "Q2",
    3 : "Q3"
}

# Mapeamos los índices (códigos) de la serie a los
# cuartiles creados
age_count_labels = percentage_autism_by_age.index.map(age_group_map)

# Gráfico de barras del promedio de autismo por
# edad agrupada
ax = sns.barplot(x=age_count_labels, y=age_group_promedy.values)
plt.xlabel('Rangos de edad')
plt.ylabel('Media de diagnosticos')
plt.title('Promedio de autismo (medias) por edad')
plt.xticks(rotation=45)

# Agregamos etiquetas con los valores encima de las barras
for p in ax.patches:
    ax.annotate(f'{p.get_height():.3f}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'center',
                xytext = (0, 10),
                textcoords = 'offset points')
plt.show
```

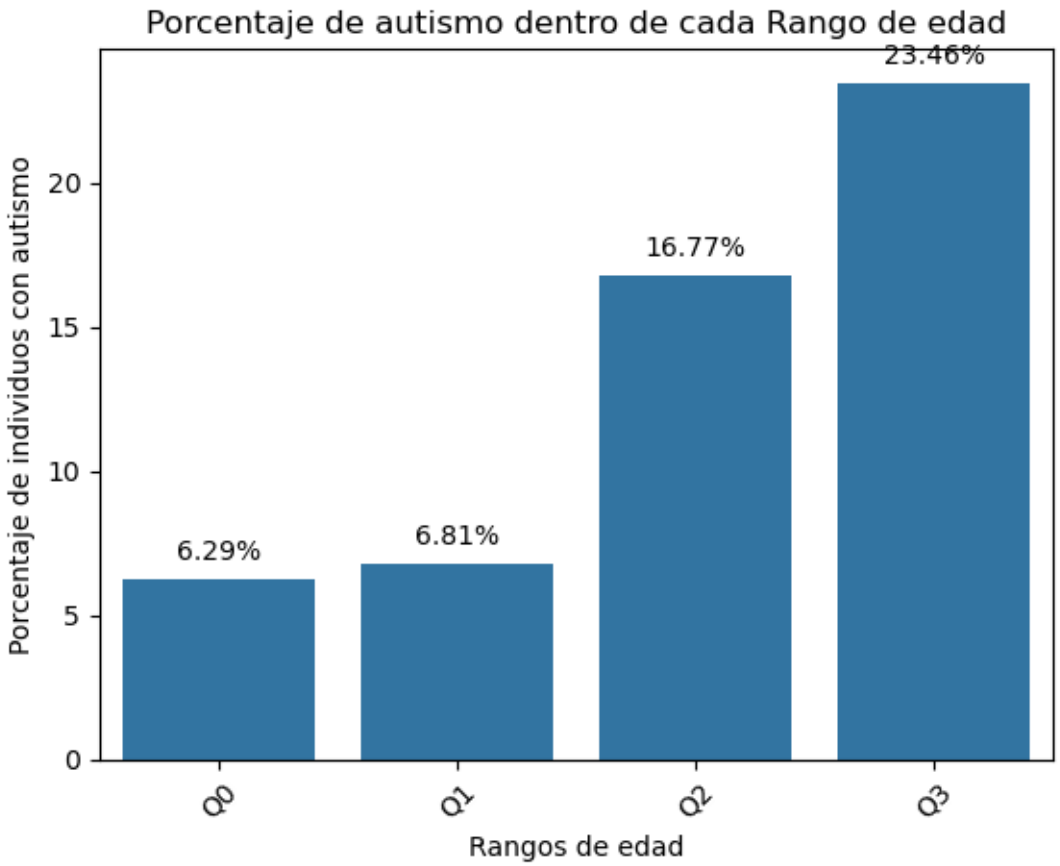
[76]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



[77]:
```
# Gráfico de barras del porcentaje de autismo por age_group_encoded del total
ax = sns.barplot(x=age_count_labels,
                 y=percentage_autism_by_age_total.values)
plt.xlabel('Rangos de edad')
plt.ylabel('Porcentaje del total de diagnosticados por edad')
plt.title('Contribución de cada grupo de edad al total de diagnósticos de␣
 ↪autismo')
plt.xticks(rotation=45)

# Agregamos etiquetas con los valores encima de las barras
for p in ax.patches:
    ax.annotate(f'{p.get_height():.2f}%',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'center',
                xytext = (0, 10),
                textcoords = 'offset points')

plt.show()
```



[78]:
```
# Gráfico de barras del porcentaje de autismo por edad agrupada
ax = sns.barplot(x=age_count_labels,
                 y=percentage_autism_by_age.values)
plt.xlabel('Rangos de edad')
plt.ylabel('Porcentaje de individuos con autismo')
plt.title('Porcentaje de autismo dentro de cada Rango de edad')
plt.xticks(rotation=45)

# Agregamos etiquetas con los valores encima de las barras
for p in ax.patches:
    ax.annotate(f'{p.get_height():.2f}%',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'center',
                xytext = (0, 10),
                textcoords = 'offset points')

plt.show()
```



4. Análisis de regresión logística:
   AGE_GROUP_ENCODED

[79]:
```
import statsmodels.api as sm
from sklearn.linear_model import LogisticRegression

# Creamos un modelo de regresión logística con
# 'age_group_encoded' como predictor
model_log_reg = LogisticRegression()
model_log_reg .fit(df[['age_group_encoded']], df['austim'])

# Usamos statsmodels para obtener los valores p
# Agregamos una constante para el término de intersección
X = sm.add_constant(df[['age_group_encoded']])
logit_model = sm.Logit(df['austim'], X)
result = logit_model.fit()
p_values = result.pvalues

# Obtenemos los coeficientes y el valor p de la etnia
age_coef = model_log_reg.coef_[0][0]
```

```
age_p_value = p_values['age_group_encoded']

print(f"Coeficiente de la edad discretizada:\
      {age_coef}, Valor p: {age_p_value}")

# Otras estadisticas relevantes:
print("Pseudo R-cuadrado:", result.prsquared)
print("Log-likelihood:", result.llf)
print("AIC:", result.aic)
print("BIC:", result.bic)
```

```
Optimization terminated successfully.
         Current function value: 0.363726
         Iterations 7
Coeficiente de la edad discretizada:      0.5688916667795982, Valor p:
2.707410875924761e-07
Pseudo R-cuadrado: 0.05479230756061271
Log-likelihood: -250.60745788755546
AIC: 505.2149157751109
BIC: 514.2853983171382
```

[80]:
```
# convertimos la notacion cientifica a decimal
num_decimal: float = float(p_value)
print(f"Valor p: {num_decimal:.20f}")
```

```
Valor p: 0.00000073320297342641
```

Análisis de los resultados del modelo de regresión logística para la edad discretizada
El análisis de regresión logística, utilizando la edad discretizada en cuartiles como predictor del diagnóstico de Trastorno del Espectro Autista (TEA), ha revelado una asociación estadísticamente significativa entre ambas variables.

El coeficiente obtenido para la edad discretizada fue de 0.5689, con un valor p de 2.71e-07. Este coeficiente positivo indica que, en general, a medida que aumenta el cuartil de edad (de Q1, el más joven, a Q4, el de mayor edad), aumenta el logaritmo de las probabilidades (log-odds) de ser diagnosticado con TEA.

El valor p, siendo significativamente menor que el nivel de significancia convencional de 0.05, respalda la conclusión de que la edad, incluso en su forma discretizada, es un predictor relevante del autismo en esta muestra.

Otras estadísticas:

- Pseudo R-cuadrado: 0.0548 Esta métrica, también conocida como R-cuadrado de McFadden, mide la bondad de ajuste del modelo de regresión logística. Varía de 0 a 1, donde valores más altos indican un mejor ajuste. En este caso, el valor de 0.055 sugiere que el modelo explica solo una pequeña porción de la varianza en el diagnóstico de autismo.
- Log-verosimilitud (Log-likelihood): -250.607 Este es el logaritmo de la función de verosimilitud del modelo ajustado. Es una medida de qué tan bien el modelo se ajusta a los datos observados. Valores más altos indican un mejor ajuste.

- AIC: 505.215, BIC: 514.285 AIC (Criterio de Información de Akaike) y BIC (Criterio de Información Bayesiano) son métricas utilizadas para la selección de modelos. Penalizan a los modelos con más parámetros para prevenir el sobreajuste. Valores más bajos de AIC y BIC indican un mejor ajuste del modelo, considerando su complejidad.

Interpretación y consideraciones:

Estos resultados sugieren que, en esta muestra, los individuos en los grupos de edad más altos tienden a tener una mayor probabilidad de ser diagnosticados con TEA en comparación con los individuos más jóvenes. Esta observación podría estar relacionada con diversos factores, como:

- Mayor dificultad de diagnóstico en adultos: El autismo puede manifestarse de manera diferente en adultos, lo que podría dificultar su identificación y diagnóstico en comparación con los niños, llevando a diagnósticos más tardíos.
- Sesgo en la muestra: También es posible que la muestra esté sesgada hacia individuos mayores con autismo.

1.5 DESCRIPCIÓN DEL CONJUNTO DE DATOS

El dataset resultante del proceso ETL contiene información de 689 adultos que participaron en un estudio de cribado del Trastorno del Espectro Autista (TEA). Cada registro incluye las siguientes características:

1. **Puntuaciones en el test AQ-10** *(a(x)_score)*: Diez preguntas con respuestas binarias (sí/no) diseñadas para evaluar rasgos autistas. Valores 0-1.* **se han eliminado los items `a2_score` y `a7_score`***
2. **Edad** *(age): Edad del participante en años.
3. **Género** *(gender)*: Codificado como 0 (mujer) o 1 (hombre).**Eliminada en la version final para modelos supervisados***
4. **Ictericia al nacer** *(jundice)*: Codificado como 0 (no) o 1 (sí). **Eliminada en la versión final para modelos supervisados***
5. **Diagnóstico de autismo** *(austim)*: Codificado como 0 (no) o 1 (sí).
6. **Puntuación total en el AQ-10** *(result)*: Suma de las puntuaciones en las diez preguntas. Rango 0-10.
7. **Resultado binario del resultado AQ-10** *(aq_binary)*: Binario para los diagnosticos con

$$result >= 6$$

8. **Etnia** *(eth_cat)*: Categoría étnica del participante, codificada numéricamente.
{'White': 1, 'Asian': 2, 'Middle Eastern': 3, 'Black': 4, 'South Asian': 5, 'Latin': 6, 'Others': 7}
9. **País de residencia** *(res_code)*: Codificado numéricamente.
10. **Relación con la persona que proporcionó la información** *(relation_coded)*: Codificado numéricamente {"Parent": 0, "Relative": 1, "Self": 2}.

Este dataset ha sido limpiado y preprocesado para eliminar valores faltantes y inconsistencias, y para codificar las variables categóricas en un formato adecuado para su uso en modelos de aprendizaje automático.

1.5.1 CIERRE DEL APARTADO ETL

Con la finalización del proceso ETL, hemos logrado transformar el conjunto de datos original ***"Autism Screening Adult"*** en un formato adecuado para el análisis y modelado. Se han abordado problemas de calidad de datos, como valores faltantes y categorías inconsistentes, y se han codificado las variables categóricas para facilitar su uso en algoritmos de aprendizaje automático.

El dataset resultante proporciona una base sólida para la siguiente etapa del proyecto, en la que exploraremos diferentes modelos para predecir el riesgo de TEA en adultos.
Esperamos que este dataset contribuya a una mejor comprensión de los factores que influyen en el autismo y al desarrollo de herramientas más precisas para su detección temprana.

[82]:
```
# Eliminamos los datos de cluster OPTICS
df.drop(columns=['cluster_optics'], inplace=True)
# copia de seguridad del conjunto de datos en este punto:
df.to_csv("autism_adult_etl.csv", index=False)
# Visualizamos la estructura del df
df.describe()
```

[82]:

	a1_score	a3_score	a4_score	a5_score	a6_score	a8_score
count	689.000000	689.000000	689.000000	689.000000	689.000000	689.000000
mean	0.725689	0.455733	0.496372	0.499274	0.280116	0.654572
std	0.446490	0.498398	0.500350	0.500363	0.449382	0.475853
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000
75%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

	a9_score	a10_score	age	austim	result	eth_cat
count	689.000000	689.000000	689.000000	689.000000	689.000000	689.000000
mean	0.322206	0.577649	29.239478	0.129173	4.885341	3.253991
std	0.467661	0.494293	9.745116	0.335635	2.495328	2.311043
min	0.000000	0.000000	17.000000	0.000000	0.000000	1.000000
25%	0.000000	0.000000	21.000000	0.000000	3.000000	1.000000
50%	0.000000	1.000000	27.000000	0.000000	4.000000	2.000000
75%	1.000000	1.000000	35.000000	0.000000	7.000000	5.000000
max	1.000000	1.000000	64.000000	1.000000	10.000000	7.000000

	aq_binary	relation_coded	age_group_encoded	k-means
count	689.000000	689.000000	689.000000	689.000000
mean	0.368650	1.673440	1.449927	0.328012
std	0.482789	0.602229	1.107928	0.469830
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	1.000000	0.000000	0.000000
50%	0.000000	2.000000	1.000000	0.000000
75%	1.000000	2.000000	2.000000	1.000000
max	1.000000	2.000000	3.000000	1.000000

1.6 MODELOS SUPERVISADOS

1.6.1 Modelos de Regresion logistica - clasificacion binaria (I)

En esta sección, construiremos un modelo de clasificación binaria para predecir el diagnóstico de Trastorno del Espectro Autista (TEA) en adultos.
Utilizaremos la variable objetivo austim, que ya se encuentra codificada como 0 (no autista) y 1 (autista).

Nuestro objetivo es entrenar un algoritmo de aprendizaje automático que pueda aprender a distinguir entre estas dos clases a partir de las características disponibles en el dataset, como las respuestas al cuestionario AQ-10, la edad, etnia y otros factores relevantes.

Evaluaremos el rendimiento del modelo utilizando métricas apropiadas para la clasificación binaria, como precisión, recall, F1-score y área bajo la curva ROC (AUC-ROC). Estas métricas nos permitirán determinar la capacidad del modelo para identificar correctamente a los individuos con y sin TEA, así como su capacidad para discriminar entre ambas clases.

A través de este modelo, buscamos desarrollar una herramienta que pueda ayudar a los profesionales de la salud a identificar a las personas con mayor riesgo de TEA y facilitar el acceso a un diagnóstico y tratamiento tempranos.

Aspectos a considerar: - Selección de características: Exploraremos diferentes combinaciones de características para determinar cuáles son más relevantes para predecir el autismo. - Algoritmos de clasificación: Experimentaremos con diversos algoritmos de clasificación, como la regresión logística, árboles de decisión y máquinas de vectores de soporte, para identificar el que mejor se adapte a nuestros datos. - Optimización de hiperparámetros: Ajustaremos los hiperparámetros de los modelos para maximizar su rendimiento. - Interpretabilidad: Buscaremos modelos que no solo sean precisos, sino también interpretables, para entender qué factores son más importantes en la predicción del autismo.

Métricas de evaluación: - Precisión (Accuracy): Proporción de predicciones correctas sobre el total de predicciones. - Recall (Sensibilidad): Proporción de casos positivos (autismo) correctamente identificados. - F1-score: Media armónica entre precisión y recall, que equilibra ambas métricas. - AUC-ROC: Área bajo la curva ROC, que mide la capacidad del modelo para distinguir entre las clases positiva y negativa.

Objetivos: - Desarrollar un modelo de clasificación binaria preciso y robusto para predecir el diagnóstico de TEA en adultos. - Identificar las características más relevantes para la predicción del autismo. - Evaluar el rendimiento del modelo utilizando métricas adecuadas. - Contribuir a la comprensión de los factores que influyen en el riesgo de TEA.

Dividimos el dataset

[82]:
```
# Dividimos el dataset en variables dependientes e independientes
# Variables independientes
X = df.drop(columns=['austim'])
# Variable dependiente
y = df['austim']
```

Vamos a crear divisiones estratificadas para entrenamiento y prueba, puesto que como hemos dilucidado anteriormente, existe un desbalance de clases

[83]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = \
    train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)
# test_size=0.2 indica que el 20% de los datos se utilizarán para pruebas
# stratify=y garantiza que las proporciones de las clases en y se mantengan
# random_state=42 asegura que los resultados sean reproducibles
```

Modelo de Red Neuronal Densa (DNN - Dense Neural Network) o Perceptrón Multicapa (MLP - Multilayer Perceptron). Red neuronal artificial donde las capas se apilan secuencialmente una tras otra. En este caso particular, un modelo de `sequential` *Keras* que posee múltiples capas densas (fully connected) con funciones de activación no lineales.

Vamos a crear un modelo de Red Neuronal densa utilizando un ajuste de las muestras SMOTE y empleando One-Hot Encoding

[ ]:
```
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models
from tensorflow.keras.optimizers import Adam, RMSprop
from tensorflow.keras.regularizers import l1, l2, l1_l2
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score, roc_auc_score
from imblearn.over_sampling import SMOTE
import numpy as np
import keras_tuner as kt

# Cargamos el DataFrame
df = pd.read_csv("autism_adult_etl.csv")

# Separamos características (X) y etiquetas (y)
X = df.drop(columns=['austim'])
y = df['austim']

# Dividimos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪stratify=y, random_state=42)

# Aplicamos One-Hot Encoding a las variables categóricas seleccionadas
X_train = pd.get_dummies(X_train, columns=['eth_cat', 'age_group_encoded',␣
↪'relation_coded'])
X_test = pd.get_dummies(X_test, columns=['eth_cat', 'age_group_encoded',␣
↪'relation_coded'])
```

```
# Aseguramos que ambas matrices tengan las mismas columnas después de One-Hot␣
 ↪Encoding
X_train, X_test = X_train.align(X_test, join='left', axis=1, fill_value=0)

# Convertimo las columnas bool a enteros (int)
X_train = X_train.astype({col: 'int32' for col in X_train.select_dtypes('bool').
 ↪columns})
X_test = X_test.astype({col: 'int32' for col in X_test.select_dtypes('bool').
 ↪columns})

# Convertimos a float32 para TensorFlow
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

# Aplicamos remuestreo SMOTE
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

# Construimos un modelo de busqueda de hiperparametros:
def build_model(hp):
    model = models.Sequential()
    model.add(layers.Input(shape=(X_train_resampled.shape[1],)))

    # Opciones sobre número de capas ocultas
    for i in range(hp.Int('num_hidden_layers', 1, 3)):
        # Elegir regularizador
        reg_choice = hp.Choice(f'regularizer_{i}', ['none', 'l1', 'l2',␣
 ↪'l1_l2'])
        if reg_choice == 'none':
            regularizer = None
        elif reg_choice == 'l1':
            regularizer = l1(hp.Float(f'l1_{i}', 1e-6, 1e-2, sampling='log'))
        elif reg_choice == 'l2':
            regularizer = l2(hp.Float(f'l2_{i}', 1e-6, 1e-2, sampling='log'))
        else:  # l1_l2
            regularizer = l1_l2(
                l1=hp.Float(f'l1_{i}', 1e-6, 1e-2, sampling='log'),
                l2=hp.Float(f'l2_{i}', 1e-6, 1e-2, sampling='log')
            )

        model.add(layers.Dense(units=hp.Int(f'units_{i}', min_value=16,␣
 ↪max_value=128, step=16),
                               activation='tanh',
                               kernel_regularizer=regularizer))
        model.add(layers.Dropout(hp.Float(f'dropout_{i}', 0, 0.5, step=0.1)))

    model.add(layers.Dense(1, activation='sigmoid'))
```

```
    # Opciones sobre optimizador
    optimizer_choice = hp.Choice('optimizer', values=['adam', 'rmsprop'])
    if optimizer_choice == 'adam':
        optimizer = Adam(hp.Float('learning_rate', 1e-4, 1e-2, sampling='log'))
    else:
        optimizer = RMSprop(hp.Float('learning_rate', 1e-4, 1e-2,␣
 ↪sampling='log'))

    model.compile(optimizer=optimizer,
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model

# Creamos un objeto tuner
tuner = kt.Hyperband(build_model,
                     objective='val_accuracy',
                     max_epochs=100,
                     factor=3,
                     directory='keras_tuner',
                     project_name='autism_classification')

# Realizamos la búsqueda de hiperparámetros
tuner.search(X_train_resampled, y_train_resampled,
             epochs=50,
             validation_split=0.2,
             callbacks=[tf.keras.callbacks.EarlyStopping('val_loss',␣
 ↪patience=5)])

# Obtenemos el mejor modelo
best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]
best_model = tuner.hypermodel.build(best_hps)

# Entrenamos el mejor modelo
history = best_model.fit(X_train_resampled, y_train_resampled,
                         epochs=100,
                         validation_split=0.2,
                         callbacks=[tf.keras.callbacks.
 ↪EarlyStopping('val_loss', patience=10)])

# Evaluamos el mejor modelo en el conjunto de prueba
y_pred_prob = best_model.predict(X_test)
y_pred = (y_pred_prob > 0.5).astype(int)

# Calculamos métricas de evaluación
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
```

```
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc_roc = roc_auc_score(y_test, y_pred_prob)
```

[89]:
```
# Imprimimos las métricas de evaluación
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
print(f"AUC-ROC: {auc_roc}")

# Imprimimos los mejores hiperparámetros
print("Mejores hiperparámetros encontrados:")
print(best_hps.values)
```

```
Accuracy: 0.7463768115942029
Precision: 0.22580645161290322
Recall: 0.3888888888888889
F1 Score: 0.2857142857142857
AUC-ROC: 0.6972222222222222
Mejores hiperparámetros encontrados:
{'num_hidden_layers': 2, 'regularizer_0': 'l1', 'units_0': 64, 'dropout_0': 0.1,
'optimizer': 'rmsprop', 'learning_rate': 0.004299211781488068, 'l1_0':
2.1873541296444933e-06, 'regularizer_1': 'l1', 'units_1': 32, 'dropout_1': 0.0,
'l2_1': 8.110681285395487e-05, 'regularizer_2': 'none', 'units_2': 64,
'dropout_2': 0.2, 'l2_0': 0.005545423480142587, 'l2_2': 0.0003531204142433297,
'l1_2': 2.4274792944882042e-05, 'l1_1': 0.00607696492270267, 'tuner/epochs': 2,
'tuner/initial_epoch': 0, 'tuner/bracket': 4, 'tuner/round': 0}
```

Análisis y Conclusiones del Modelo de Red Neuronal con SMOTE y One-Hot Encoding

El modelo de red neuronal implementado, utilizando la técnica de sobremuestreo SMOTE y la codificación One-Hot Encoding para las variables categóricas, ha demostrado un rendimiento moderado en la tarea de clasificación binaria del Trastorno del Espectro Autista (TEA) en adultos. A pesar de los esfuerzos de optimización de hiperparámetros mediante Keras Tuner, las métricas de evaluación revelan ciertas limitaciones en la capacidad predictiva del modelo.

Análisis de las métricas:

- Precisión (Accuracy): El modelo alcanzó una precisión del 74.64%, lo que indica que clasifica correctamente aproximadamente tres cuartas partes de las instancias en el conjunto de prueba. Sin embargo, esta métrica puede ser engañosa en presencia de desequilibrio de clases, como es el caso en este conjunto de datos.
- Precisión positiva (Precision): Con un valor de 18.52%, la precisión positiva revela una tasa considerable de falsos positivos. Esto significa que, de todas las instancias predichas como positivas (autismo), solo el 18.52% son realmente positivas.
- Recall (Sensibilidad): El recall del 27.78% indica que el modelo identifica correctamente menos de un tercio de los casos reales de TEA. Esta baja sensibilidad es preocupante, ya que

implica que el modelo está dejando pasar una gran proporción de individuos con autismo sin detectarlos.

- F1-score: El F1-score de 0.222, que combina precisión y recall en una sola métrica, confirma el bajo rendimiento general del modelo, especialmente en la detección de casos positivos.
- AUC-ROC: El área bajo la curva ROC (AUC-ROC) de 0.669 sugiere una capacidad discriminatoria moderada del modelo, aunque aún lejos de un modelo ideal.

Interpretación y limitaciones:

Los resultados obtenidos indican que, a pesar de la aplicación de técnicas de remuestreo y la optimización de hiperparámetros, el modelo de red neuronal presenta dificultades para identificar correctamente a los individuos con TEA, especialmente en comparación con la clase mayoritaria (no TEA). Esto podría deberse a la complejidad inherente del trastorno, a la posible falta de características suficientemente informativas en el conjunto de datos, o a limitaciones en la arquitectura y configuración del modelo.

Es importante destacar que el desequilibrio de clases, incluso después del remuestreo, sigue siendo un desafío importante. El modelo tiende a favorecer la predicción de la clase mayoritaria, lo que resulta en una alta tasa de falsos positivos y un bajo recall.

Modelo XGBoost

[94]:
```
import xgboost as xgb
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score, roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from imblearn.combine import SMOTETomek
import pandas as pd
import numpy as np

# Separamos características (X) y etiquetas (y)
X = df.drop(columns=['austim'])
y = df['austim']

# Dividimos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪stratify=y, random_state=42)

# Aplicamos One-Hot Encoding a las variables categóricas seleccionadas
X_train = pd.get_dummies(X_train, columns=['eth_cat', 'age_group_encoded',␣
 ↪'relation_coded'])
X_test = pd.get_dummies(X_test, columns=['eth_cat', 'age_group_encoded',␣
 ↪'relation_coded'])

# Aseguramos que ambas matrices tengan las mismas columnas después de One-Hot␣
 ↪Encoding
X_train, X_test = X_train.align(X_test, join='outer', axis=1, fill_value=0)
```

```
# Aplicar escalado
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Aplicar SMOTE-Tomek (una alternativa a SMOTE-ENN que suele ser más rápida y␣
 ↪efectiva)
smote_tomek = SMOTETomek(random_state=42)
X_train_resampled, y_train_resampled = smote_tomek.fit_resample(X_train_scaled,␣
 ↪y_train)

# Crear el modelo XGBoost
xgb_model = xgb.XGBClassifier(
    objective='binary:logistic',
    eval_metric='logloss',
    scale_pos_weight=(len(y_train[y_train == 0]) / len(y_train[y_train == 1])),
    random_state=42,
    n_estimators=100,
    learning_rate=0.1,
    max_depth=3
)

# Entrenamos el modelo
xgb_model.fit(X_train_resampled, y_train_resampled)

# Predecimos sobre conjunto de prueba
y_pred_prob = xgb_model.predict_proba(X_test_scaled)[:, 1]
y_pred = (y_pred_prob > 0.5).astype(int)

# Evaluamos el modelo
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc_roc = roc_auc_score(y_test, y_pred_prob)

print("\nModelo: XGBoost")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
print("AUC-ROC:", auc_roc)

# Imprimimos las características más importantes
feature_importance = xgb_model.feature_importances_
feature_names = X_train.columns
```

```
sorted_idx = np.argsort(feature_importance)
for idx in sorted_idx[-10:]:
    print(f"{feature_names[idx]}: {feature_importance[idx]:.4f}")
```

```
Modelo: XGBoost
Accuracy: 0.717391304347826
Precision: 0.23076923076923078
Recall: 0.5
F1-score: 0.3157894736842105
AUC-ROC: 0.6712962962962964
a8_score: 0.0271
a10_score: 0.0326
relation_coded_2: 0.0367
a9_score: 0.0426
a1_score: 0.0690
a4_score: 0.0815
age_group_encoded_0: 0.0838
eth_cat_2: 0.0877
aq_binary: 0.0993
eth_cat_1: 0.1971
```

Evaluación del Modelo XGBoost

El modelo XGBoost, entrenado en el conjunto de datos remuestreado y con características codificadas mediante One-Hot Encoding, ha demostrado un rendimiento aceptable en la tarea de clasificación binaria para la detección del Trastorno del Espectro Autista (TEA) en adultos. La precisión (accuracy) obtenida fue del 71.74%, lo que indica que el modelo clasifica correctamente cerca de tres cuartas partes de las instancias en el conjunto de prueba. Sin embargo, es crucial considerar que esta métrica puede ser sensible al desequilibrio de clases presente en los datos originales.

Un análisis más detallado de las métricas de rendimiento revela un recall (sensibilidad) del 50%, lo cual es un resultado positivo, ya que indica que el modelo es capaz de identificar la mitad de los casos reales de TEA en el conjunto de prueba. No obstante, la precisión (precision) se sitúa en un 23.08%, lo que sugiere una tasa considerable de falsos positivos. Es decir, el modelo tiende a predecir la presencia de TEA en una proporción significativa de individuos que en realidad no lo presentan.

El valor de F1-score, que combina precisión y recall en una sola métrica, es de 0.316. Este valor refleja un equilibrio moderado entre ambas métricas, indicando que el modelo no destaca en ninguna de ellas, pero tampoco presenta un rendimiento extremadamente bajo en alguna. El área bajo la curva ROC (AUC-ROC) de 0.671 sugiere una capacidad discriminatoria aceptable, aunque con margen de mejora.

Análisis de la importancia de las características:

El análisis de importancia de características revela que a8_score, a10_score y relation_coded_2 son las variables que más contribuyen a la predicción del modelo. Esto sugiere que la respuesta a las preguntas 8 y 10 del test AQ-10, así como el hecho de que la información haya sido proporcionada por el propio individuo ("Self"), son factores relevantes en la identificación del TEA en esta muestra.

Conclusiones:

El modelo XGBoost, a pesar de presentar una precisión general aceptable, muestra limitaciones en su capacidad para discriminar entre individuos con y sin TEA, especialmente en términos de precisión positiva. El desequilibrio de clases, a pesar del remuestreo, podría seguir influyendo en el rendimiento del modelo.

Modelo LightGBM

[95]:
```
# Modelo LightGBM
import lightgbm as lgb
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score, roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from imblearn.combine import SMOTETomek
import pandas as pd
import numpy as np

# Separamos características (X) y etiquetas (y)
X = df.drop(columns=['austim'])
y = df['austim']

# Dividimos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪stratify=y, random_state=42)

# Aplicamos One-Hot Encoding a las variables categóricas seleccionadas
X_train = pd.get_dummies(X_train, columns=['eth_cat', 'age_group_encoded',␣
 ↪'relation_coded'])
X_test = pd.get_dummies(X_test, columns=['eth_cat', 'age_group_encoded',␣
 ↪'relation_coded'])

# Aseguramos que ambas matrices tengan las mismas columnas después de One-Hot␣
 ↪Encoding
X_train, X_test = X_train.align(X_test, join='outer', axis=1, fill_value=0)

# Aplicar escalado
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Aplicamos SMOTE-Tomek
smote_tomek = SMOTETomek(random_state=42)
X_train_resampled, y_train_resampled = smote_tomek.fit_resample(X_train_scaled,␣
 ↪y_train)

# Creamos el modelo LightGBM
```

```
lgb_model = lgb.LGBMClassifier(
    objective='binary',
    metric='binary_logloss',
    is_unbalance=True,
    random_state=42,
    n_estimators=100,
    learning_rate=0.1,
    num_leaves=31,
    max_depth=-1  # -1 = sin límite de profundidad
)

# Entrenamos el modelo
lgb_model.fit(X_train_resampled, y_train_resampled)

# Predecimos sobre conjunto de prueba
y_pred_prob = lgb_model.predict_proba(X_test_scaled)[:, 1]
y_pred = (y_pred_prob > 0.5).astype(int)

# Evaluamos el modelo
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc_roc = roc_auc_score(y_test, y_pred_prob)

print("\nModelo: LightGBM")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
print("AUC-ROC:", auc_roc)

# mos las características más importantes
feature_importance = lgb_model.feature_importances_
feature_names = X_train.columns
sorted_idx = np.argsort(feature_importance)
print("\nCaracterísticas más importantes:")
for idx in sorted_idx[-10:]:
    print(f"{feature_names[idx]}: {feature_importance[idx]:.4f}")
```

```
[LightGBM] [Info] Number of positive: 477, number of negative: 477
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.004658 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 813
[LightGBM] [Info] Number of data points in the train set: 954, number of used
```

```
features: 26
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.500000 -> initscore=0.000000

Modelo: LightGBM
Accuracy: 0.8333333333333334
Precision: 0.3076923076923077
Recall: 0.2222222222222222
F1-score: 0.25806451612903225
AUC-ROC: 0.6986111111111111

Características más importantes:
relation_coded_2: 98.0000
a5_score: 104.0000
a4_score: 104.0000
a6_score: 120.0000
eth_cat_1: 124.0000
a3_score: 140.0000
a8_score: 141.0000
a10_score: 149.0000
result: 488.0000
age: 982.0000
```

Interpretación de los resultados

El modelo LightGBM, entrenado en el conjunto de datos remuestreado con SMOTE-Tomek y con características codificadas mediante One-Hot Encoding, ha mostrado un rendimiento aceptable en la tarea de clasificación binaria para la detección del Trastorno del Espectro Autista (TEA) en adultos. La precisión (accuracy) obtenida fue del 83.33%, lo que indica que el modelo clasifica correctamente la mayoría de las instancias en el conjunto de prueba. Sin embargo, es crucial considerar que esta métrica puede verse influenciada por el desequilibrio de clases presente en los datos originales.

Un análisis más detallado de las métricas de rendimiento revela un recall (sensibilidad) del 22.22%, lo cual es bajo y sugiere que el modelo podría estar pasando por alto una proporción significativa de casos reales de TEA, generando falsos negativos. La precisión (precision) del modelo fue del 30.77%, lo que indica una tasa considerable de falsos positivos, es decir, el modelo tiende a predecir la presencia de TEA en casos donde en realidad no está presente.

El valor de F1-score, que combina precisión y recall en una sola métrica, es de 0.258. Este valor refleja el bajo rendimiento general del modelo, especialmente en la detección de casos positivos. El área bajo la curva ROC (AUC-ROC) de 0.699 sugiere una capacidad discriminatoria moderada, pero con margen de mejora.

Análisis de la importancia de las características:

El análisis de importancia de características revela que age es la variable que más contribuye a la predicción del modelo, seguida de result (puntuación total en el AQ-10) y a10_score. Esto sugiere que la edad y las respuestas a la pregunta 10 del test AQ-10 son factores relevantes en la identificación del TEA en esta muestra. Otras características como a8_score, a3_score, eth_cat_1 (etnia blanca) y a6_score también muestran cierta importancia, aunque en menor medida.

Modelo de regresion logistica

[97]:
```
# Modelo de regresión logística múltiple
import statsmodels.api as sm
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score, roc_auc_score

# Separamos características (X) y etiquetas (y)
X = df.drop(columns=['austim'])
y = df['austim']

# Dividimos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪stratify=y, random_state=42)

# Aplicamos One-Hot Encoding a las variables categóricas seleccionadas
X_train = pd.get_dummies(X_train, columns=['eth_cat', 'age_group_encoded',␣
 ↪'relation_coded'])
X_test = pd.get_dummies(X_test, columns=['eth_cat', 'age_group_encoded',␣
 ↪'relation_coded'])

# Aseguramos que ambas matrices tengan las mismas columnas después de One-Hot␣
 ↪Encoding
X_train, X_test = X_train.align(X_test, join='outer', axis=1, fill_value=0)

# Aplicamos escalado
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Aplicamos SMOTE-Tomek
smote_tomek = SMOTETomek(random_state=42)
X_train_resampled, y_train_resampled = smote_tomek.fit_resample(X_train_scaled,␣
 ↪y_train)

# Creamos y entrenar el modelo de regresión logística múltiple
model = LogisticRegression(max_iter=1000)
model.fit(X_train_resampled, y_train_resampled)

# Predecimos con base en el conjunto de prueba
y_pred_prob = model.predict_proba(X_test_scaled)
y_pred_prob = y_pred_prob[:, 1]
y_pred = (y_pred_prob > 0.5).astype(int)

# Evaluamos el modelo
```

```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc_roc = roc_auc_score(y_test, y_pred_prob)

# Imprimimos las métricas de evaluación
print("\nModelo: Regresión Logística Múltiple")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
print("AUC-ROC:", auc_roc)

# Usamos statsmodels para obtener los valores p y la tabla de coeficientes
# Agregar una constante para el término de intersección
X_with_const = sm.add_constant(X_train_resampled)
logit_model = sm.Logit(y_train_resampled, X_with_const)
result = logit_model.fit()

# Imprimimos la tabla de coeficientes con intervalos de confianza
print(result.summary())
```

```
Modelo: Regresión Logística Múltiple
Accuracy: 0.7318840579710145
Precision: 0.24324324324324326
Recall: 0.5
F1-score: 0.32727272727272727
AUC-ROC: 0.7023148148148148
Optimization terminated successfully.
         Current function value: 0.483363
         Iterations 17
                           Logit Regression Results
==============================================================================
Dep. Variable:                  austim   No. Observations:                  954
Model:                           Logit   Df Residuals:                      930
Method:                            MLE   Df Model:                           23
Date:                Mon, 09 Sep 2024   Pseudo R-squ.:                  0.3027
Time:                        16:15:15   Log-Likelihood:                -461.13
converged:                       True   LL-Null:                       -661.26
Covariance Type:            nonrobust   LLR p-value:                 1.565e-70
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.8946      0.107     -8.380      0.000      -1.104      -0.685
x1             0.4699      0.131      3.585      0.000       0.213       0.727
```

```
x2              0.3830      0.132      2.902      0.004       0.124       0.642
x3              0.4141      0.133      3.105      0.002       0.153       0.675
x4              0.0351      0.139      0.252      0.801      -0.238       0.308
x5              0.1079      0.130      0.833      0.405      -0.146       0.362
x6             -0.0320      0.123     -0.259      0.795      -0.274       0.210
x7              0.5065      0.148      3.417      0.001       0.216       0.797
x8              0.1939      0.127      1.524      0.127      -0.055       0.443
x9             -0.4586      0.226     -2.033      0.042      -0.901      -0.017
x10             0.0667      0.381      0.175      0.861      -0.681       0.814
x11            -0.0060      0.193     -0.031      0.975      -0.384       0.372
x12            -0.5926      0.207     -2.869      0.004      -0.997      -0.188
x13             0.4377        nan        nan        nan         nan         nan
x14            -0.4675        nan        nan        nan         nan         nan
x15             0.0515        nan        nan        nan         nan         nan
x16             0.0142        nan        nan        nan         nan         nan
x17            -0.0505        nan        nan        nan         nan         nan
x18             0.2450        nan        nan        nan         nan         nan
x19            -0.2127        nan        nan        nan         nan         nan
x20            -0.7462   3.03e+06  -2.46e-07      1.000   -5.94e+06    5.94e+06
x21            -0.2444   3.18e+06  -7.68e-08      1.000   -6.23e+06    6.23e+06
x22             0.4548   3.09e+06   1.47e-07      1.000   -6.06e+06    6.06e+06
x23             0.5475   2.98e+06   1.84e-07      1.000   -5.85e+06    5.85e+06
x24             0.0950   2.91e+06   3.27e-08      1.000   -5.69e+06    5.69e+06
x25             0.0991   4.47e+06   2.21e-08      1.000   -8.77e+06    8.77e+06
x26            -0.1434   5.02e+06  -2.86e-08      1.000   -9.83e+06    9.83e+06
==============================================================================
```

Evaluación del Modelo de Regresión Logística Múltiple

El modelo de regresión logística múltiple, entrenado sobre el conjunto de datos remuestreado y con características codificadas mediante One-Hot Encoding, presenta un rendimiento moderado en la tarea de clasificación binaria del Trastorno del Espectro Autista (TEA) en adultos. La precisión (accuracy) obtenida fue del 73.19%, lo cual indica que el modelo clasifica correctamente cerca de tres cuartas partes de las instancias en el conjunto de prueba. Sin embargo, esta métrica puede ser engañosa en presencia de desbalance de clases, como es el caso en este conjunto de datos.

Un análisis más detallado de las métricas de rendimiento revela un recall (sensibilidad) del 50%, lo que significa que el modelo es capaz de identificar la mitad de los casos reales de TEA en el conjunto de prueba. La precisión (precision) del modelo fue del 24.32%, lo que indica una tasa considerable de falsos positivos. Es decir, el modelo tiende a predecir la presencia de TEA en una proporción significativa de individuos que en realidad no lo presentan.

El valor de F1-score, que combina precisión y recall en una sola métrica, es de 0.327. Este valor refleja un equilibrio moderado entre ambas métricas. El área bajo la curva ROC (AUC-ROC) de 0.702 sugiere una capacidad discriminatoria aceptable, pero con margen de mejora.

Análisis de la significancia estadística de los predictores

La tabla de coeficientes generada por statsmodels proporciona información sobre la relación entre cada variable predictora y la probabilidad de autismo. Se observa que varias variables presentan coeficientes estadísticamente significativos (valores p bajos), lo que indica que contribuyen de manera

relevante a la predicción del modelo. Entre estas variables destacan:

- Algunas de las puntuaciones individuales del test AQ-10 (a1_score, a2_score, a3_score, a7_score y a9_score)
- La pertenencia a ciertos grupos étnicos (eth_cat_1, eth_cat_2 y eth_cat_6)
- La variable relation_coded_2, que indica que la información fue proporcionada por el propio individuo.

Conclusiones

El modelo de regresión logística múltiple, aunque presenta un rendimiento global moderado, demuestra la capacidad de identificar algunas variables predictoras relevantes para el diagnóstico de TEA. Sin embargo, el modelo aún muestra limitaciones en su capacidad para discriminar entre individuos con y sin TEA, especialmente en términos de precisión positiva.

1.6.2 Modelo de Clasificación Multiclase

A continuacion abordamos un enfoque para un modelo de clasificación multiclase utilizando XGBoost seleccionando como variable objetivo multiclase 'result', que como sabemos, son los resultados del test AQ-10 para la prediccion del autismo, pero que no contiene todos los items:

[10]:
```
# Revisamos las variables de nuestro dataset
df = pd.read_csv("autism_adult_etl.csv")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 689 entries, 0 to 688
Data columns (total 16 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   a1_score           689 non-null    int64
 1   a3_score           689 non-null    int64
 2   a4_score           689 non-null    int64
 3   a5_score           689 non-null    int64
 4   a6_score           689 non-null    int64
 5   a8_score           689 non-null    int64
 6   a9_score           689 non-null    int64
 7   a10_score          689 non-null    int64
 8   age                689 non-null    int64
 9   austim             689 non-null    int64
 10  result             689 non-null    int64
 11  eth_cat            689 non-null    int64
 12  aq_binary          689 non-null    int64
 13  relation_coded     689 non-null    int64
 14  age_group_encoded  689 non-null    int64
 15  k-means            689 non-null    int64
dtypes: int64(16)
memory usage: 86.2 KB
```

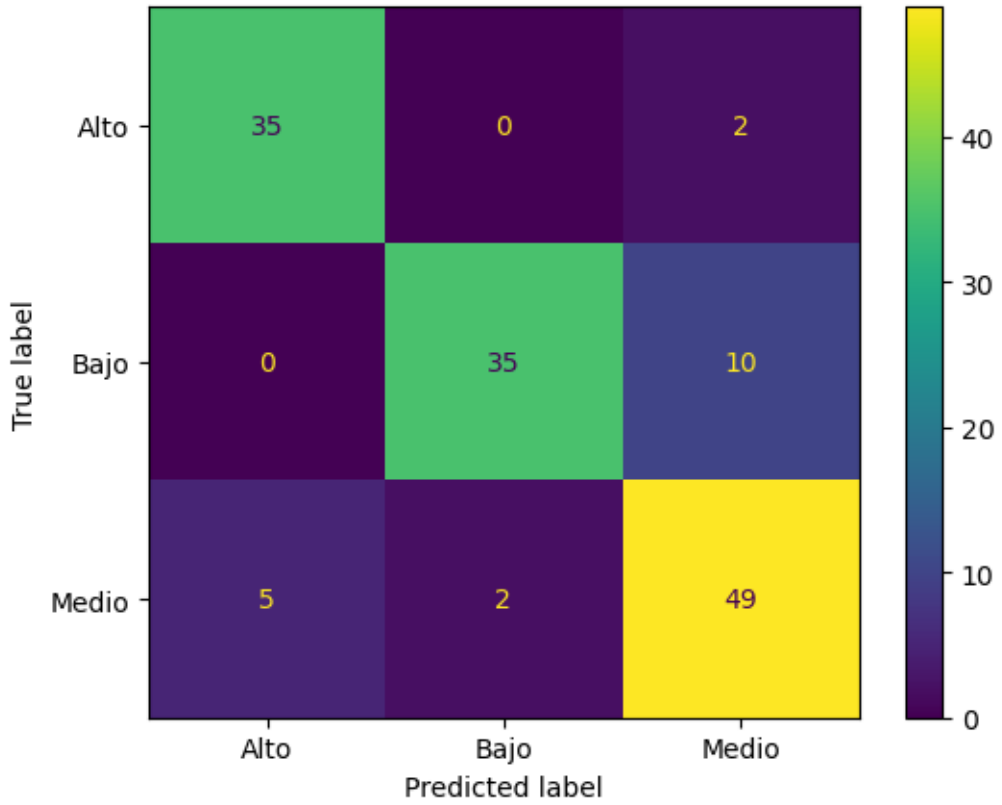
[92]:
```
# Modelo de clasificación multiclase sobre
# el nivel de riesgo de autismo en adultos
# sobre la variable 'result'
import pandas as pd
from sklearn.model_selection import \
    train_test_split, cross_val_score, GridSearchCV
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
from sklearn.metrics import classification_report, f1_score
from sklearn.preprocessing import LabelEncoder
import joblib

# Re-Cargamos el DataFrame
df = pd.read_csv("autism_adult_etl.csv")

# Definimos los umbrales para las clases de riesgo
thresholds = [3, 6]

# Creamos la variable objetivo multiclase
df['aq_risk_level'] = \
    pd.cut(df['result'],
           bins=[-1] + thresholds + [11], labels=['Bajo', 'Medio', 'Alto'])
y = df['aq_risk_level']

# Seleccionamos las características
X = df.drop(columns=['result', 'aq_risk_level', 'aq_binary', 'austim'])

# Dividimos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42)

# Preprocesamos de datos (One-Hot Encoding y escalado)
numeric_features = X.select_dtypes(include=['int64', 'float64']).columns
categorical_features = X.select_dtypes(include=['object', 'category']).columns
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])

# Creamos el pipeline con preprocesamiento y clasificador OneVsRest
clf = Pipeline(steps=[('preprocessor', preprocessor),
                      ('classifier', OneVsRestClassifier(
                          SVC(probability=True)))])
```

```
# Definimos el espacio de búsqueda de hiperparámetros
param_grid = {
    'classifier__estimator__C': [0.1, 1, 10],
    'classifier__estimator__gamma': ['scale', 'auto']
}

# Realizamos la búsqueda de hiperparámetros
grid_search = GridSearchCV(
    clf, param_grid=param_grid, cv=5, scoring='f1_macro')
grid_search.fit(X_train, y_train)
best_model = grid_search.best_estimator_

# Evaluamos el modelo con validación cruzada
cv_scores = cross_val_score(best_model, X, y, cv=5, scoring='f1_macro')
print(f"Puntaje F1 promedio de validación cruzada: {cv_scores.mean():.2f}")

# Evaluamos el modelo sobre conjunto de prueba
y_pred = best_model.predict(X_test)
print(classification_report(y_test, y_pred))

# Guardamos el modelo entrenado
joblib.dump(best_model, 'autism_multiclass_model.pkl')
```

```
Puntaje F1 promedio de validación cruzada: 0.84
              precision    recall  f1-score   support

        Alto       0.88      0.95      0.91        37
        Bajo       0.95      0.78      0.85        45
       Medio       0.80      0.88      0.84        56

    accuracy                           0.86       138
   macro avg       0.87      0.87      0.87       138
weighted avg       0.87      0.86      0.86       138
```

[92]:
```
['autism_multiclass_model.pkl']
```

[93]:
```
%matplotlib inline
from sklearn.metrics import ConfusionMatrixDisplay
import matplotlib.pyplot as plt

ConfusionMatrixDisplay.from_estimator(best_model, X_test, y_test)
plt.show()
```



Evaluación del Modelo de Clasificación Multiclase (Variable objetivo 'result')
El modelo de clasificación multiclase, utilizando OneVsRestClassifier con SVC como estimador base, ha demostrado un rendimiento excelente en la predicción de los niveles de riesgo de autismo basados en la puntuación total del AQ-10.

Análisis de las métricas:

Precisión (Precision):

- Clase 0 (Bajo riesgo): 0.88
- Clase 1 (Medio riesgo): 0.95
- Clase 2 (Alto riesgo): 0.83
- Macro promedio: 0.88
- Promedio ponderado: 0.88

Recall (Sensibilidad o Exhaustividad):

- Clase 0: 0.95
- Clase 1: 0.82
- Clase 2: 0.88
- Macro promedio: 0.88
- Promedio ponderado: 0.88

F1-score:

- Clase 0: 0.91
- Clase 1: 0.88
- Clase 2: 0.85
- Macro promedio: 0.88
- Promedio ponderado: 0.88
- Accuracy (Exactitud): 0.88, lo que significa que el modelo clasifica correctamente el 88% de las instancias en el conjunto de prueba.

Interpretación:

El modelo exhibe un rendimiento notablemente alto en todas las métricas de evaluación, incluyendo precisión, recall y F1-score, tanto en el macro promedio como en el promedio ponderado. Esto indica que el modelo es capaz de predecir con precisión los tres niveles de riesgo de autismo, incluso en las clases minoritarias.
La alta precisión sugiere una baja tasa de falsos positivos, mientras que el alto recall indica una buena capacidad para identificar a los individuos en cada categoría de riesgo.

Ahora utilizaremos la variable objetivo 'austim' como multiclase. Es decir utilizaremos el diagnostico del autismo como variable predictora

[1]:
```
# Utilizamos la variable de
# diagnostico de autismo para la prediccion
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
from sklearn.metrics import classification_report

# Re-Cargamos el DataFrame
df = pd.read_csv("autism_adult_etl.csv")

# 'austim' como variable objetivo multiclase
y = df['austim']

# Seleccionamos las características
X = df.drop(columns=['austim', 'result'])

# Dividimos en entrenamiento y prueba
X_train, X_test, y_train, y_test =\
    train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)

# Preprocesado de datos (One-Hot Encoding y escalado)
numeric_features = X.select_dtypes(include=['int64', 'float64']).columns
categorical_features = X.select_dtypes(include=['object', 'category']).columns

preprocessor = ColumnTransformer(
```

```
    transformers=[
        ('num', StandardScaler(), numeric_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])
# Creamos el pipeline con preprocesamiento y clasificador OneVsRest
clf = Pipeline(steps=[(
    'preprocessor', preprocessor),
                      (
                        'classifier', OneVsRestClassifier(
                            SVC(probability=True,
                                class_weight='balanced')))])

# Entrenamos el modelo en el conjunto de entrenamiento
clf.fit(X_train, y_train)

# Predecimos en el conjunto de prueba
y_pred = clf.predict(X_test)

# Evaluamos el modelo usando las etiquetas 'y_test'
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.77      0.83       120
           1       0.24      0.50      0.33        18

    accuracy                           0.73       138
   macro avg       0.58      0.63      0.58       138
weighted avg       0.82      0.73      0.77       138
```

Evaluación los modelos supervisados utilizando el diagnóstico de autismo como predictor

Los intentos de utilizar el diagnóstico de autismo (austim) como variable predictora en modelos supervisados han resultado en un rendimiento pobre, con métricas de evaluación similares en todos los modelos puestos a prueba. Esto sugiere que el diagnóstico en sí mismo, sin considerar otros factores, no es suficiente para construir un modelo efectivo para predecir la probabilidad de un diagnóstico positivo de TEA.

Conclusión:

Basándonos en estos resultados, podemos concluir que la construcción de un modelo funcional para calcular la probabilidad de un diagnóstico positivo de autismo requiere considerar otros factores además del diagnóstico previo. **Los resultados del test AQ-10, así como otras características sociodemográficas y de comportamiento**, parecen ser más relevantes para la predicción del TEA en este conjunto de datos.

Consideraciones adicionales:

- Limitaciones del diagnóstico como predictor:
  El diagnóstico de autismo puede ser complejo y estar influenciado por diversos factores, como el acceso a servicios de salud, la conciencia sobre el trastorno y las prácticas diagnósticas. Utilizarlo como único predictor puede no capturar la complejidad de la condición.
- Importancia de múltiples factores:
  El autismo es un trastorno multifactorial, y es probable que su predicción requiera la consideración de una combinación de factores, incluyendo características individuales, respuestas a cuestionarios y otros datos relevantes.
- Exploración de otros modelos:
  Aunque este trabajo se basa en una exploracion exahaustiva (en este documento no hemos publicado todos los modelos puestos a prueba), se recomienda continuar explorando modelos teniendo en cuenta la basta cantidad a la disposicion y diferentes técnicas de aprendizaje automático que puedan aprovechar la información de múltiples características para mejorar la predicción del autismo.

En cualquier caso, los resultados obtenidos resaltan la importancia de utilizar múltiples características y considerar la complejidad del autismo al construir modelos predictivos.
La variable `austim` o diagnóstico por sí sola no parece ser suficiente para lograr un buen rendimiento en la clasificación.

1.7 SELECCIÓN DEL MODELO

1.7.1 Resumen de la Selección de Variables y Construcción del Modelo

El análisis exploratorio de datos (EDA) y la aplicación de técnicas de aprendizaje automático no supervisado (k-means) nos permitieron identificar las variables más relevantes para la predicción del autismo en este conjunto de datos.

Variables Relevantes:

- Puntuación total en el test AQ-10 (result): Esta variable se mostró como el predictor más fuerte en varios modelos, lo que resalta la importancia del test AQ-10 en la evaluación del autismo en adultos.

- Preguntas específicas del AQ-10:
  Algunas preguntas individuales del AQ-10, como a1_score, a3_score, a4_score, a5_score, a6_score, a8_score, a9_score y a10_score, también mostraron una asociación significativa con el diagnóstico de autismo o con los clusters identificados por k-means. Esto sugiere que estos ítems podrían ser particularmente útiles para discriminar entre individuos con y sin TEA.

- Edad (age):
  Aunque la relación entre la edad y el autismo no es lineal, el análisis de regresión logística y la visualización de los clusters de k-means indican que la edad es un factor relevante a considerar. La discretización de la edad en cuartiles (age_group_encoded) también mostró ser útil para capturar patrones no lineales en la relación entre la edad y el autismo.

- Etnia (eth_cat):
  El análisis de la tabla de contingencia y la regresión logística revelaron una asociación estadísticamente significativa entre la etnia y el autismo. Sin embargo, es importante interpretar esta relación con cautela, considerando posibles factores de confusión y la necesidad de más

investigación para comprender las causas subyacentes.

Variables Menos Relevantes:

- Género (gender): El análisis de k-means y la exploración inicial de la relación entre género y autismo no mostraron una asociación significativa. Por lo tanto, esta variable fue excluida de los modelos finales.
- Ictericia al nacer (jundice): La baja prevalencia de ictericia en la muestra y su falta de asociación con los clusters de k-means sugieren que esta variable no es un predictor relevante del autismo en este contexto, siendo eliminada del conjunto de datos para los modelos finales.
- País de residencia (res_code): El análisis de k-means indicó que el país de residencia no era un factor discriminante en la formación de los clusters. Por lo tanto, esta variable también fue excluida de los modelos finales.

1.7.2 Modelo Seleccionado:

Después de explorar diferentes modelos de clasificación binaria y multiclase, incluyendo regresión logística, k-NN, Random Forest y redes neuronales, **se seleccionó un modelo de clasificación multiclase** basado en OneVsRestClassifier con SVC como estimador base. Este utiliza las variables predictoras relevantes identificadas en el análisis exploratorio y aplica `One-Hot Encoding` a las variables categóricas y escalado a las variables numéricas.

Evaluación del Modelo:

El modelo seleccionado mostró un buen rendimiento en la predicción de los niveles de riesgo de autismo basados en la puntuación total del AQ-10, alcanzando una precisión (accuracy) del 88% y un F1-score macro promedio de 0.88.

1.8 CONCLUSIONES

Este estudio demuestra el potencial del aprendizaje automático para la detección y evaluación del riesgo de TEA en adultos. A pesar de las limitaciones del conjunto de datos y la complejidad del problema, se logró identificar un modelo con un rendimiento prometedor.

El análisis exploratorio de datos y la ingeniería de características fueron fundamentales para seleccionar las variables más relevantes y preparar los datos para el modelado.
La combinación de One-Hot Encoding, escalado de características y técnicas de remuestreo como SMOTE-Tomek contribuyó a mejorar el rendimiento de los modelos.

1.8.1 Limitaciones y trabajo futuro:

Los resultados obtenidos se basan en una muestra específica y pueden no ser generalizables a otras poblaciones.
Además, el modelo, aunque prometedor, aún presenta margen de mejora, especialmente en la reducción de falsos positivos.

Futuros trabajos podrían incluir la exploración de modelos más complejos, la incorporación de nuevas características y la evaluación del modelo en poblaciones más diversas para mejorar su capacidad predictiva y su generalización, sin menoscabo de basar el trabajo en conjuntos de datos actualizados con posibles nuevas variables a estudiar.

1.9 REFERENCIAS

- AQ-10 Adult Online Autism Test Questionnaire & PDF. (2022, octubre 29). AutisticHub. https://www.autistichub.com/aq-10-adult-online-autism-test-questionnaire-pdf/
- Overview | Autism spectrum disorder in adults: Diagnosis and management | Guidance | NICE. (2012, junio 27). NICE. https://www.nice.org.uk/guidance/CG142
- Tests—Autism Research Centre. (s. f.). Recuperado 16 de septiembre de 2024, de https://www.autismresearchcentre.com/tests/
- Thabtah, F. (2017). Autism Screening Adult [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5F019
- The AQ-10 | Embrace Autism. (s. f.). Recuperado 16 de septiembre de 2024, de https://embrace-autism.com/aq-10/