Autism Spectrum Disorder (ASD) Prediction in Adults: A Machine Learning Approach

September 16, 2024

1 Study on Autism Spectrum Disorder (ASD) in adults

1.1 INDEX

1.2 INTRODUCTION

Autism Spectrum Disorder (ASD) is a neurodevelopmental condition affecting communication, social interaction, and behavior. Early and accurate detection of ASD in adults is crucial to provide them with the necessary support and interventions to improve their quality of life.

This project aims to investigate the contributing factors and develop machine learning models capable of predicting the risk of ASD in adults based on demographic data and responses to a screening questionnaire. We will utilize the "Autism Screening Adult" dataset from the UCI Machine Learning repository UCI Machine Learning which comprises information on 704 adults, including their scores on the Autism Spectrum Quotient (AQ) test and a clinical diagnosis of ASD.

Our approach will focus on data preparation and exploratory data analysis (EDA), followed by a multi-model strategy. We will adapt the original dataset to create optimized datasets for each specific type of machine learning model:

1. Binary Classification Model: This model will focus on predicting whether an individual has ASD or not, simplifying the problem to a two-class classification.
2. Multiclass Classification Model: This model will go beyond binary classification, attempting to predict different levels of ASD risk, potentially offering a more nuanced and personalized assessment.

For each model, we will select and train specific algorithms, evaluating their performance and comparing their results. We will explore techniques such as logistic regression, decision trees, and support vector machines, aiming to identify the most accurate and suitable model for each approach.

In addition to developing predictive models, this project seeks to contribute to the understanding of the **Factors that influence the risk of ASD in adults**.
We will analyze the importance of different features and how they relate to the diagnosis of ASD, which could provide valuable insights for future research and intervention strategies.

Keywords:
Autism Spectrum Disorder, machine learning, binary classification, multiclass classification, AQ test, diagnosis, prediction.

1.3 ETL (Extraction, transformation, load)

1.3.1 Data Extraction and Initial Loading

The data source for this project is the "Autism Screening Adult" dataset, publicly available from the UCI Machine Learning repository. The data was provided in ARFF (Attribute-Relation File Format), a common format in data mining that allows for storing metadata alongside the data.

To facilitate its manipulation and analysis in Python, the data was loaded from the ARFF file into a Pandas DataFrame, a widely-used tabular data structure in data science. This initial extraction and loading step enabled us to access the data and initiate the transformation and exploration process.

[ ]:
```
# importamos las librerias
import tensorflow as tf
import pandas as pd
import keras as kr
```

```
import scipy as scp
import numpy as np
```

Loading the dataset, originally in `.arff` format:

[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 Data Preprocessing

Since the dataset may contain missing or inconsistent values, it's essential to perform cleaning and preprocessing before proceeding with analysis and modeling. Pandas, a powerful data analysis library in Python, offers efficient tools for this task.

Data Loading

We begin by loading the data from the preprocessed CSV file into a Pandas DataFrame, providing us with a flexible tabular structure for data manipulation and exploration.

[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 TRANSFORMATION

Data Transformation In this critical phase, a series of operations and techniques were applied to clean, modify, and prepare the data extracted from the original ARFF file. The primary goal of this stage was to ensure data quality, consistency, and suitability for subsequent analysis and machine learning modeling.

Common data quality issues, such as missing values and inconsistent categories, were addressed. Additionally, encoding strategies were implemented to transform categorical variables into a numerical format suitable for machine learning algorithms. Feature engineering also played a significant role in this phase, enabling the creation of new variables and the transformation of existing ones to enhance their predictive power.

Finally, resampling techniques were applied to address the class imbalance present in the target variable, ensuring a more equitable representation of the different categories in the dataset.

[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
```

We eliminated missing values in the variable age ('age'). Since there was a minimal percentage (0.28%) of missing values (2 records), we proceeded to eliminate the corresponding rows:

[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
```

We notice that two records have been deleted. We proceed to clean the field or attribute names by converting them to lowercase and normalizing them:

[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')
```

The AQ-10 Questionnaire and its Representation in the Dataset

The Autism-Spectrum Quotient (AQ-10) is a 10-item self-report questionnaire designed to assess autistic traits in adults. Each question offers response options ranging from "Definitely Agree" to "Definitely Disagree," with each response encoded as a numerical value. The sum of these individual scores provides a total score, which is interpreted as an indicator of the likelihood of an individual exhibiting autistic traits.

In the dataset employed, the attributes "A1_Score," "A2_Score," etc., represent the scores obtained for each of the 10 AQ-10 questions. As these scores are inherently numerical, any columns that were loaded as 'object' or 'float' data types were converted to an integer (int) type to ensure their proper utilization in subsequent analyses.

[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
```

We examined the age column aiming to identify any potential outliers or extreme values.

[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
```

An identified outlier in the age attribute was removed. Subsequently, this attribute was converted to an integer data type.

[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
```

We reviewed the attributes again:

[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')
```

Subsequently, we will analyzed the gender field, which comprises two categories: “male” and “female.”

[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']
```

The gender column will be encoded as follows: 0 for “Female” and 1 for “Male”.

Next, we will analyze the categorical variable ethnicity. We will begin by examining the distribution of its values within the 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
```

The presence of the character '?' was observed in the ethnicity column, indicating missing or unknown data. It was decided to group these values into the 'Others' category to simplify the analysis. Additionally, an inconsistency was identified in the 'others' category, which was standardized to 'Others' to maintain data consistency.

Subsequently, further transformations were performed on the categories within the ethnicity variable.

[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
```

Some categories seemed redundant, such as 'Hispanic' and 'Latino', so we proceeded to group them:

[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
```

We included the categories with fewer records 'Pasifika' and 'Turkish' in 'Others' to avoid problems with the model:

[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()
```

We mapped the categories within the 'ethnicity' attribute by creating a dictionary:

[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
```

When creating ML models, the categorical variable 'ethnicity' seemed to have no usefulness, so we proceeded to eliminate it:

[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
```

It was deemed necessary to convert the AQ-10 test results into a binary format to facilitate the identification of potential autism cases. According to the test criteria, a score of 6 or higher indicates the need for a specialized evaluation, whereas a score below 6 suggests a low probability of ASD.

Consequently, a new binary variable called aq_binary was created, where scores equal to or greater than 6 were encoded as True (1), and scores below 6 as False (0). This transformation will allow this variable to be used as a simplified indicator of autism risk in subsequent analyses.

[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
```

The attribute 'jundice', indicating whether the subject was born with jaundice, was explored. An analysis was conducted to determine if any association existed between the presence of jaundice at birth and the diagnosis of autism within the sample.

[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']
```

We observe that, indeed, **'no'** has been encoded as 0 and **'yes'** as 1, respectively. We verified if the changes have indeed occurred:

[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
```

The target variable austim, representing the autism diagnosis, was encoded.

[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']
```

The encoding of the austim variable was established, assigning the value 0 to the "no" category and the value 1 to the "yes" category.

Subsequently, the distribution of the target variable austim was analyzed to assess the balance between classes (presence or absence of autism diagnosis) and to verify that the attribute type was suitable for use in machine learning models.

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

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

Although the data transformation yielded promising results, a class imbalance was identified in the sample, necessitating the implementation of resampling techniques during model training. The decision was made to utilize either SMOTE (Synthetic Minority Over-sampling Technique) or ADASYN (Adaptive Synthetic Sampling), both capable of generating synthetic samples of the minority class to balance the dataset.

The next attribute examined was country_of_res, which refers to the participant's country of residence. Although initially considered potentially redundant due to the inclusion of ethnicity in the analysis, it was decided to retain this attribute temporarily to explore its relationship with both ethnicity and the autism diagnosis during the exploratory data analysis phase.

To examine whether there was any significant association between these variables, techniques such as contingency tables, chi-squared tests, and visualizations were employed.

The following presents a detailed analysis of the content of this field.

[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
```

Given the presence of 67 different countries in the dataset, some with a very small number of samples, encoding techniques such as `One-Hot Encoding` and `Mean Encoding` were considered. However, taking into account the need to obtain a compact dataset suitable for the application of various machine learning models, it was decided to encode each country using `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}
```

We proceeded to export the data for later queries to a JSON format:

[32]:
```python
# 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]:
```python
# 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

Once the countries are encoded, we can remove the 'country_of_res' attribute

[34]:
```python
# 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
```

The next attribute considered for analysis was used_app_before, which indicated whether the participant had previously used any autism-related applications. The relevance of this attribute was evaluated within the context of the study, taking into account the following considerations:

- Familiarity with the topic: Prior use of an application could suggest greater knowledge about autism or an active search for related information or support.
- Potential selection bias: Application usage could indicate differences between participants who used them and those who did not, potentially introducing bias into the results.
- Potential effect of the application: If the application used was an intervention tool, its prior use could be associated with changes in the participant's behavior or symptoms.

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

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

Given the small number of "yes" cases and the potential for bias we found, we proceeded to eliminate the "yes" cases at this early stage of the analysis and the variable. This will allow us to build more robust and generalizable models based on a more homogeneous group of subjects.

[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
```

We proceeded to convert the `result` attribute to an integer:

[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
```

Next we explored the 'relation' variable which refers to who provided the information for the record:

[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
```

Let's group the different relationships for better model performance: - values '?' to 'Health care professional' - grouping of small categories

[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
```

Following the category grouping, a more balanced distribution was observed in the relation variable, reflecting a greater involvement of relatives and a lesser presence of parents in the data collection process. This finding aligns with the nature of the dataset, which focuses on adults diagnosed with autism, where parental involvement tends to be lower compared to studies in younger populations.

Subsequently, the relation attribute was encoded for inclusion in the machine learning models.

[40]:
```python
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]:
```python
# 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
```

Finally, the age_desc attribute was discarded, as information regarding the participants' age was already available in the numerical variable age.

Regarding `class_asc`, since no information was found in the UCI repository documentation and it only contained binary values ("yes"/"no"), the decision was made to remove it from the dataset.

[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
```

Saved a backup:

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

Exploratory Data Analysis (EDA) and Data Quality Assurance (DQA) - Phase I Following established convention, Exploratory Data Analysis (EDA) and Data Quality Assurance (DQA) were conducted as an integral part of the Transformation phase within the ETL process. However, it is worth noting that these activities can also be performed independently, subsequent to the initial ETL stage.

To carry out EDA and DQA, various built-in functions from the Pandas library were employed, with the aim of obtaining descriptive statistics that would allow for an initial understanding of the data.

[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

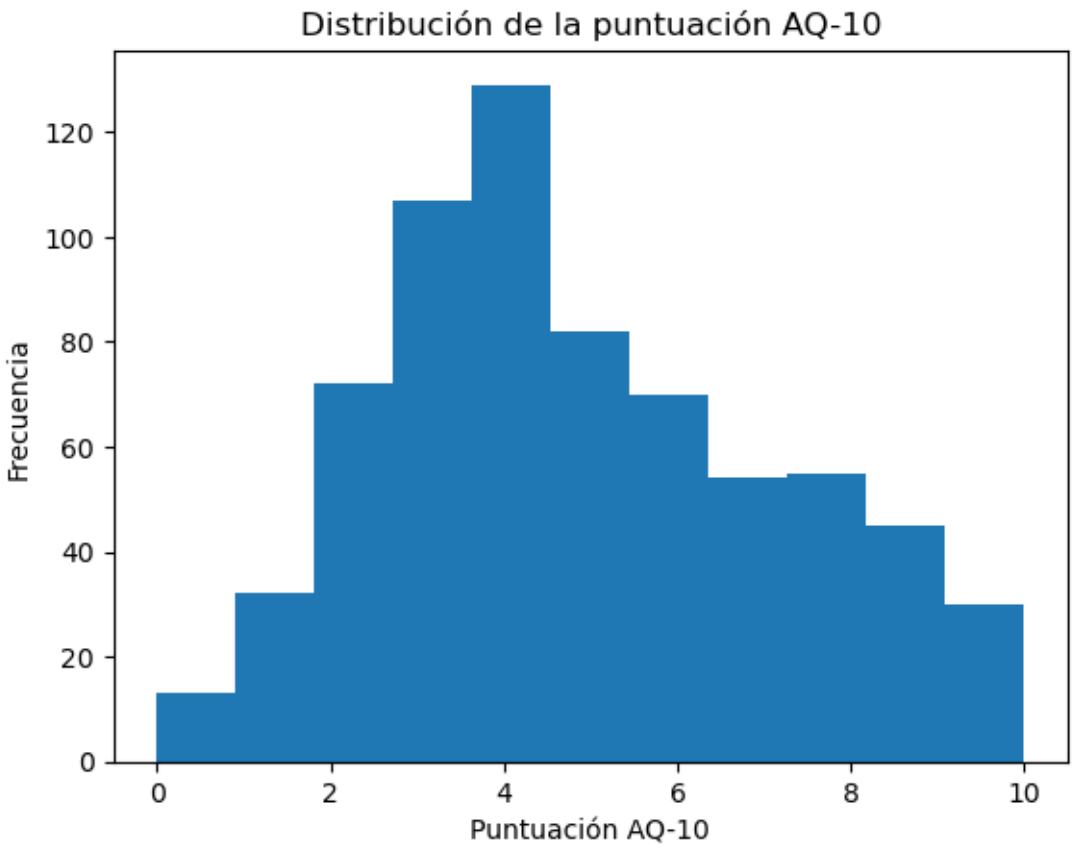
	jundice	austim	result	eth_cat	aq_binary	res_code
count	689.000000	689.000000	689.000000	689.000000	689.000000	689.000000
mean	0.097242	0.129173	4.885341	3.253991	0.368650	44.095791
std	0.296503	0.335635	2.495328	2.311043	0.482789	19.263008
min	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000
25%	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
```

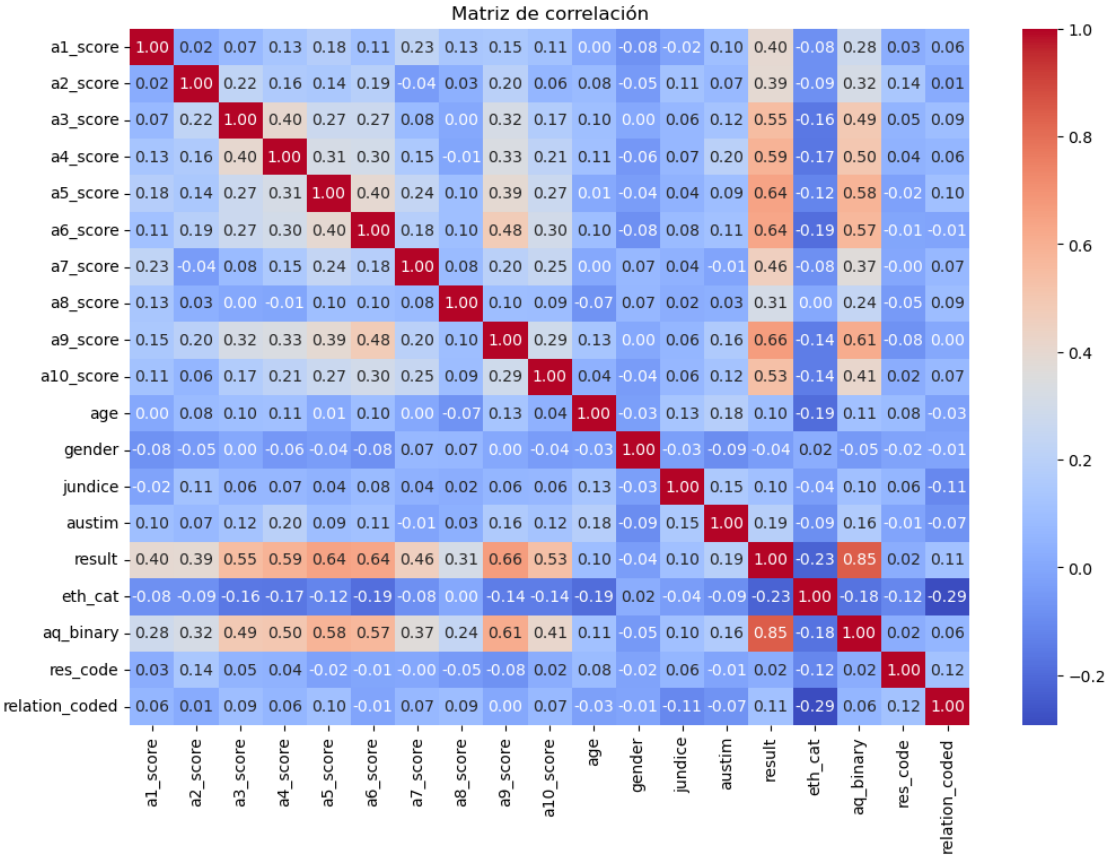
[45]:
```python
# visualizamos la asimetria positiva de 'result
plt.hist(df['result'], bins=11)  # 11 bins para cada valor posible de 0 a 10
plt.xlabel('Puntuación AQ-10')
plt.ylabel('Frecuencia')
plt.title('Distribución de la puntuación AQ-10')
plt.show()
```



[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(…)`

The exploratory data analysis revealed several key characteristics:

Distribution of AQ-10 test responses: Responses to individual AQ-10 test questions were relatively uniformly distributed, with a mean close to 0.5 and a similar standard deviation. This suggests no clear tendency towards affirmative or negative responses in any of the questions.

Demographic characteristics: The majority of participants were young adults, with a mean age of 29.2 years. A slight gender imbalance was observed, with more males (52.4%) than females (47.6%). The prevalence of jaundice at birth and autism was low (9.7% and 12.9%, respectively).

AQ-10 total score: The distribution of the total score on the AQ-10 (result) showed a positive skewness, indicating that most participants had low scores, with few cases of high scores. This is consistent with the low prevalence of autism in the sample.

Ethnicity and country of residence: Most participants were of White ethnicity and resided in the United States. However, the dataset included a diversity of ethnicities and countries of residence, which could be useful for exploring potential cultural or geographical differences in the manifestation of autism.

Relationship with the information provider: Most responses were provided by the participants themselves, increasing the reliability of the data.

Correlation Matrix:

The analysis of the correlation matrix did not reveal strong linear correlations between the variables. This suggests that the prediction of autism may depend on a complex combination of factors rather than a single dominant variable.

Despite the lack of strong linear correlations, no variables were discarded at this stage. Non-linear relationships and potential interactions between variables will be explored as we progress in model development.

Next Steps and Considerations

In the following sections, we will build and evaluate various machine learning models: binary classification - linear regression, and multiclass classification supported by unsupervised models. Cross-validation and appropriate evaluation metrics will be used to compare model performance and select the most suitable one for predicting the risk of ASD in adults.

Most importantly, we will extract insights, if any, on how the variables influence the diagnosis of autism.

It is important to acknowledge that this dataset presents a class imbalance, with a higher proportion of individuals without autism. This imbalance will be addressed during model training and evaluation, using techniques like resampling or class weighting to ensure fair and accurate performance. However, it is also acknowledged that the sample and the variables considered may not be sufficient for creating a perfect model, highlighting the complexity of ASD diagnosis and the need for further research.

1.4 Unsupervised Models

Unsupervised models were explored to gain a deeper understanding of the data. This exploration aimed to:

- Uncover underlying patterns in the data that are not apparent in a superficial analysis.
- Generate new features that could improve the predictive ability of supervised models.
- Gain a better understanding of the data structure and relationships between variables.
- Potentially enhance the performance of classification models.

1.4.1 Feature Engineering

Age Discretization Since age, as a continuous numerical variable, could exhibit a non-linear relationship with the autism diagnosis, it was decided to discretize it into intervals to capture potential non-linear patterns and improve the predictive capability of the models.

Age was divided into quartiles, i.e., four equal-sized groups based on its distribution in the dataset. This strategy allowed for the exploration of potential age effects at different stages of adulthood without assuming a predefined linear relationship.

The qcut function from Pandas was used to perform the discretization, assigning descriptive labels to each quartile (Q1, Q2, Q3, Q4). Subsequently, Label Encoding was applied to convert these labels into ordinal numerical values, facilitating their use in machine learning models.

Justification for the choice:

The division into quartiles was considered appropriate due to the lack of a specific hypothesis about the relationship between age and autism. In the absence of prior information about relevant cut-off points, quartiles allowed for a more flexible exploration of the age variable and the possibility of discovering patterns not evident in its continuous form.

Considerations:

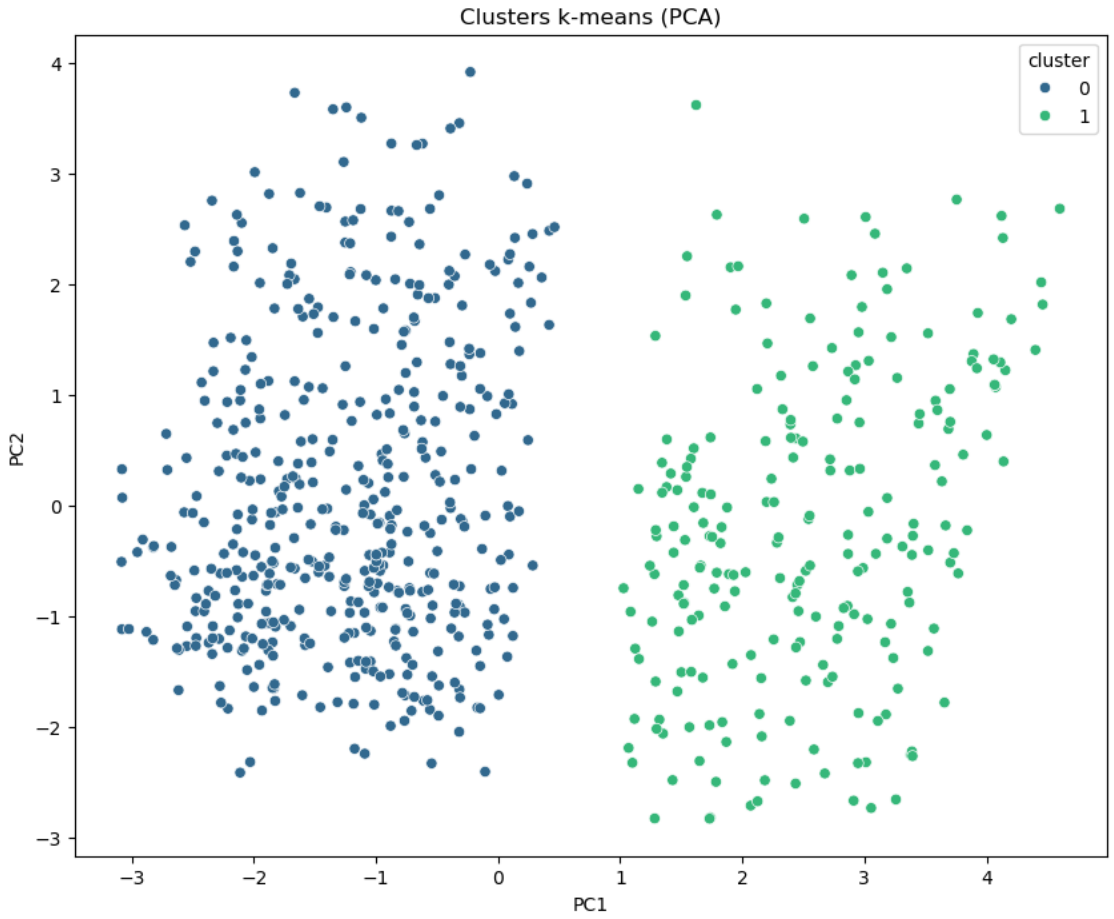
- Loss of information: Discretizing age implies a loss of granularity in the information. However, this loss is compensated by the possibility of capturing non-linear relationships and improving the interpretability of the results. -Impact assessment: The impact of this transformation on model performance will be evaluated in subsequent stages of the project.

K-Means Model

[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
```



Evaluation of K-means Clustering

The results of the k-means clustering, visualized using PCA and evaluated with internal metrics, suggested a moderately defined cluster structure in the data.

- Silhouette Score: The value of 0.188 indicated a fair separation between the clusters, with some degree of overlap. This suggested that the clusters had some internal coherence and were distinguishable, although not very markedly.
- Davies-Bouldin Index: The value of 1.98 supported the previous observation. This index, which measures the similarity between clusters, showed a moderate level of separation between the groups, indicating that some clusters might be close or overlapping.
- Calinski-Harabasz Index: With a value of 150.29, this index suggested a decent clustering quality. As it favors dense and well-separated clusters, the obtained value pointed towards a reasonable cluster structure.

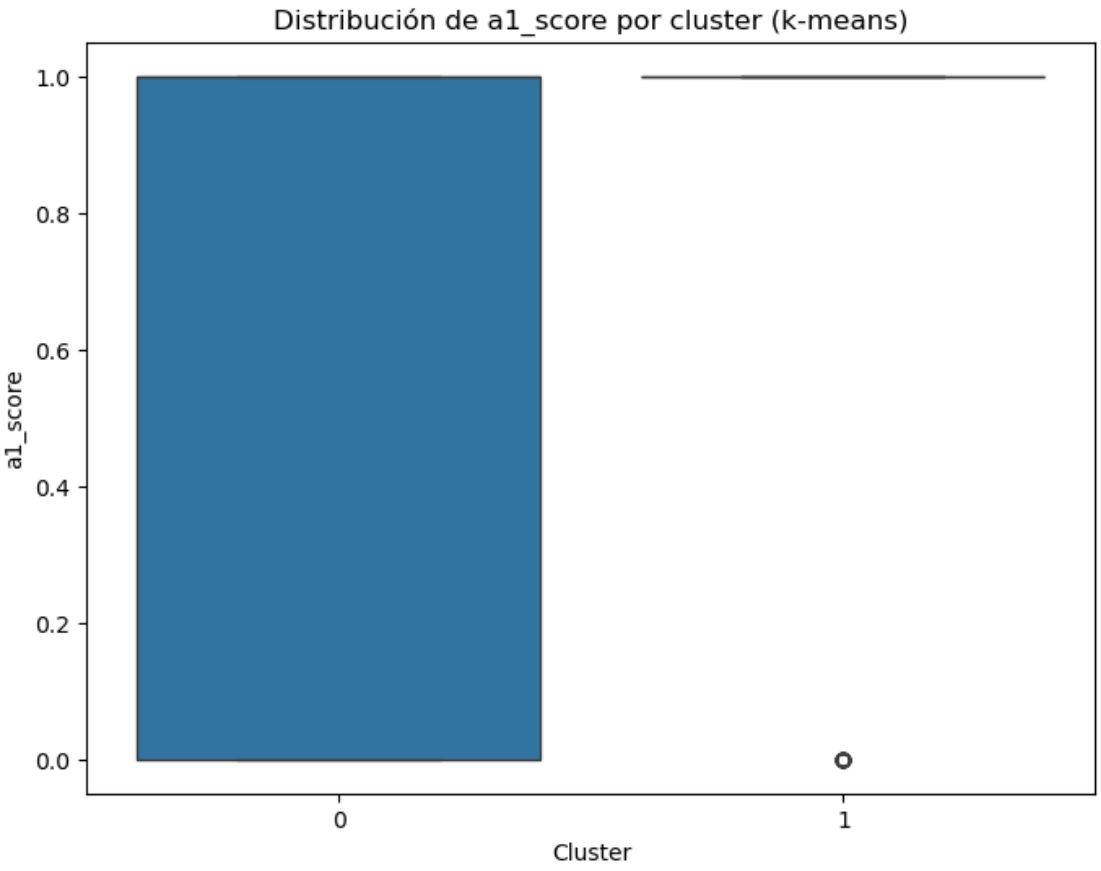
Scatter Plot (PCA):

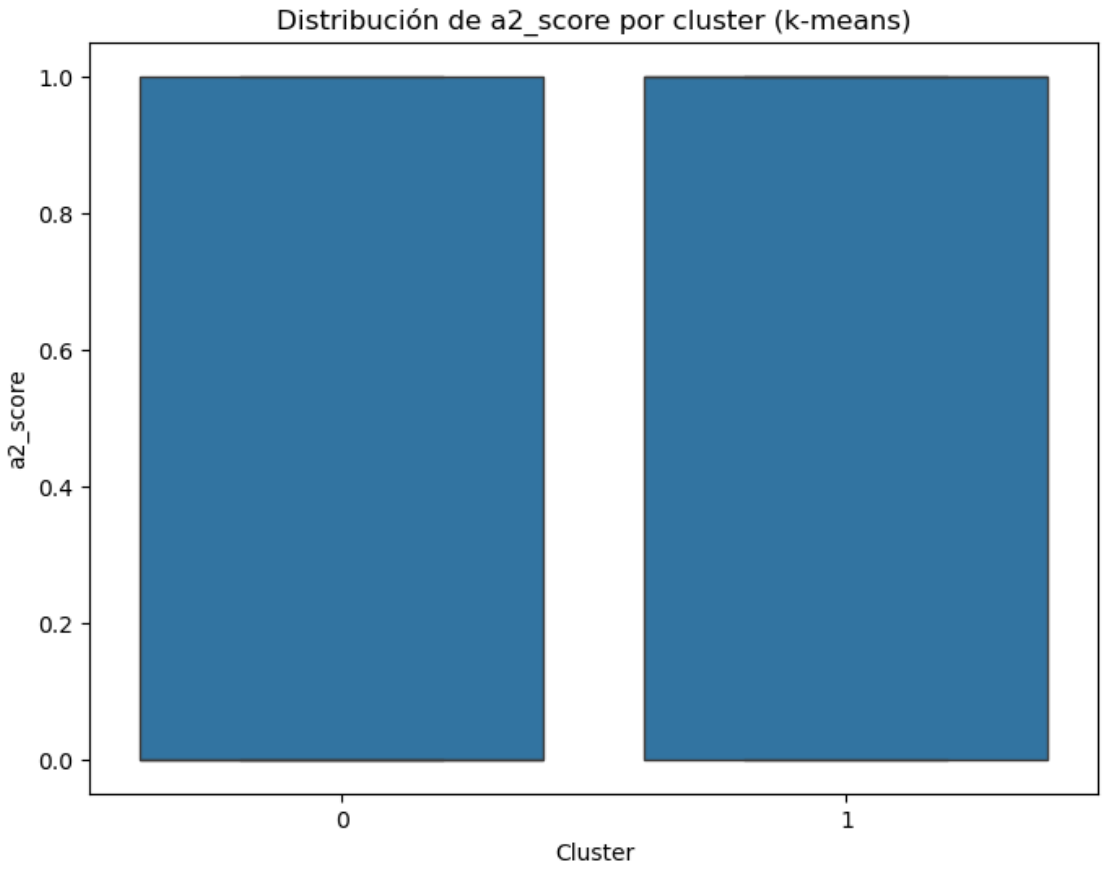
The visualization of the clusters in the space of the first two principal components showed a partial separation between the two groups. While there was a tendency for the clusters to group in different regions of the plot, there was also considerable overlap, especially in the central area. This

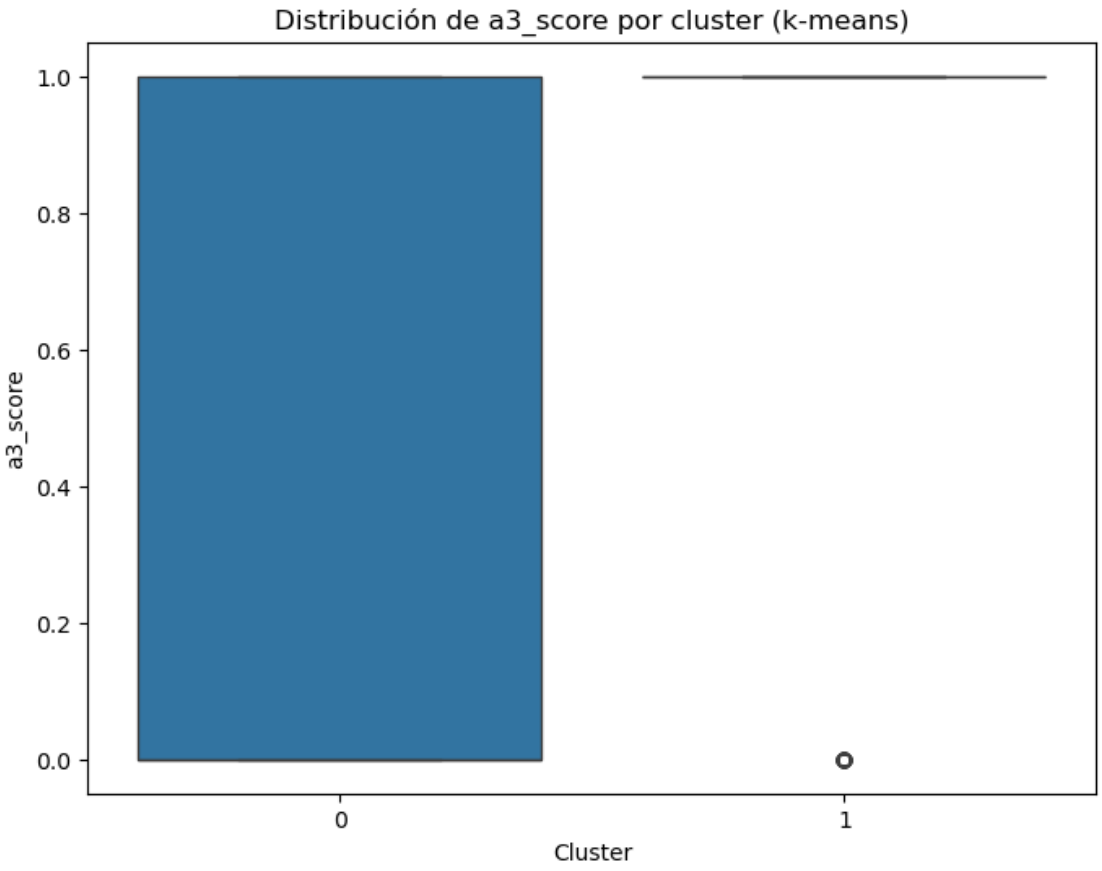
confirmed the interpretation of the internal metrics, indicating that the clusters were not completely separated.

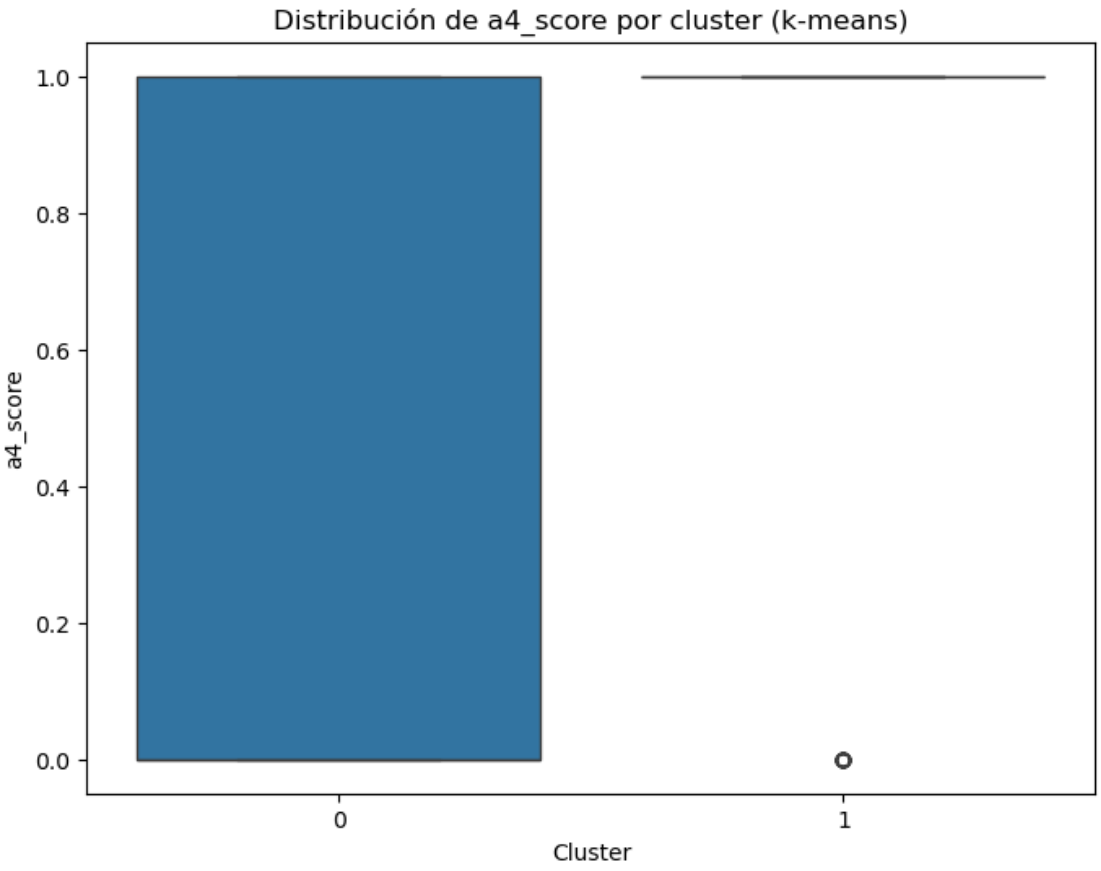
Overall, the k-means algorithm was able to identify two clusters in the data, but the separation between them was not perfect. This could be attributed to the nature of the data, the choice of features, or the inherent complexity of the autism detection problem.

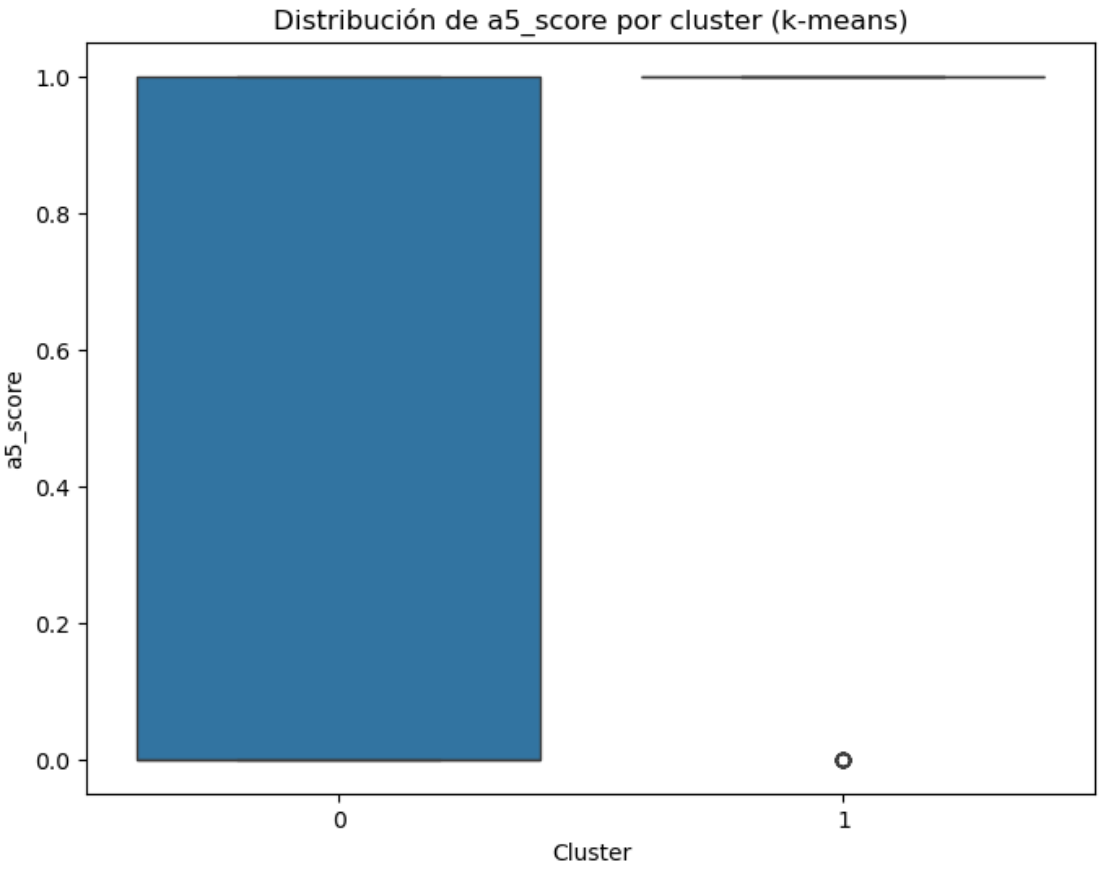
[59]:
```
# Graficamos la distribución de las
# variables a los clusters:
import seaborn as sns
import matplotlib.pyplot as plt

# 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()
```

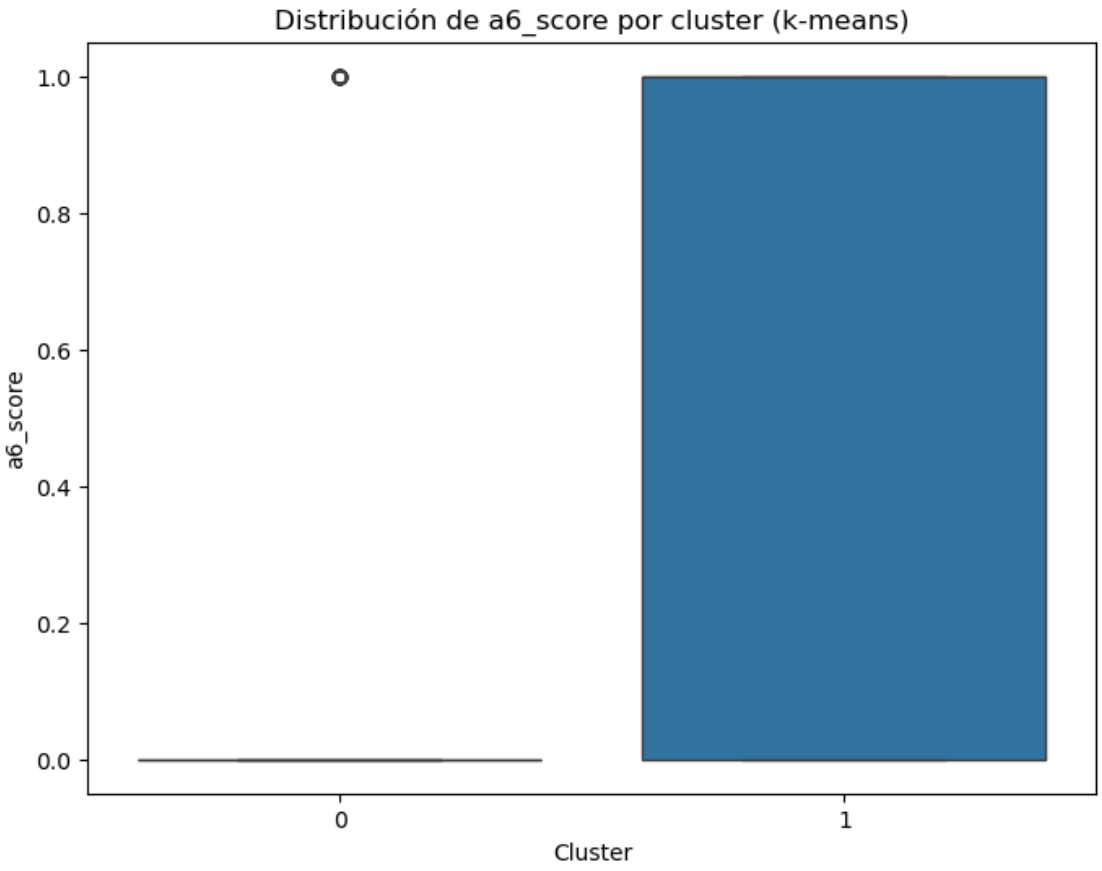

Distribución de a1_score por cluster (k-means)
a1_score
1.0
0.8
0.6
0.4
0.2
0.0
0
1
Cluster

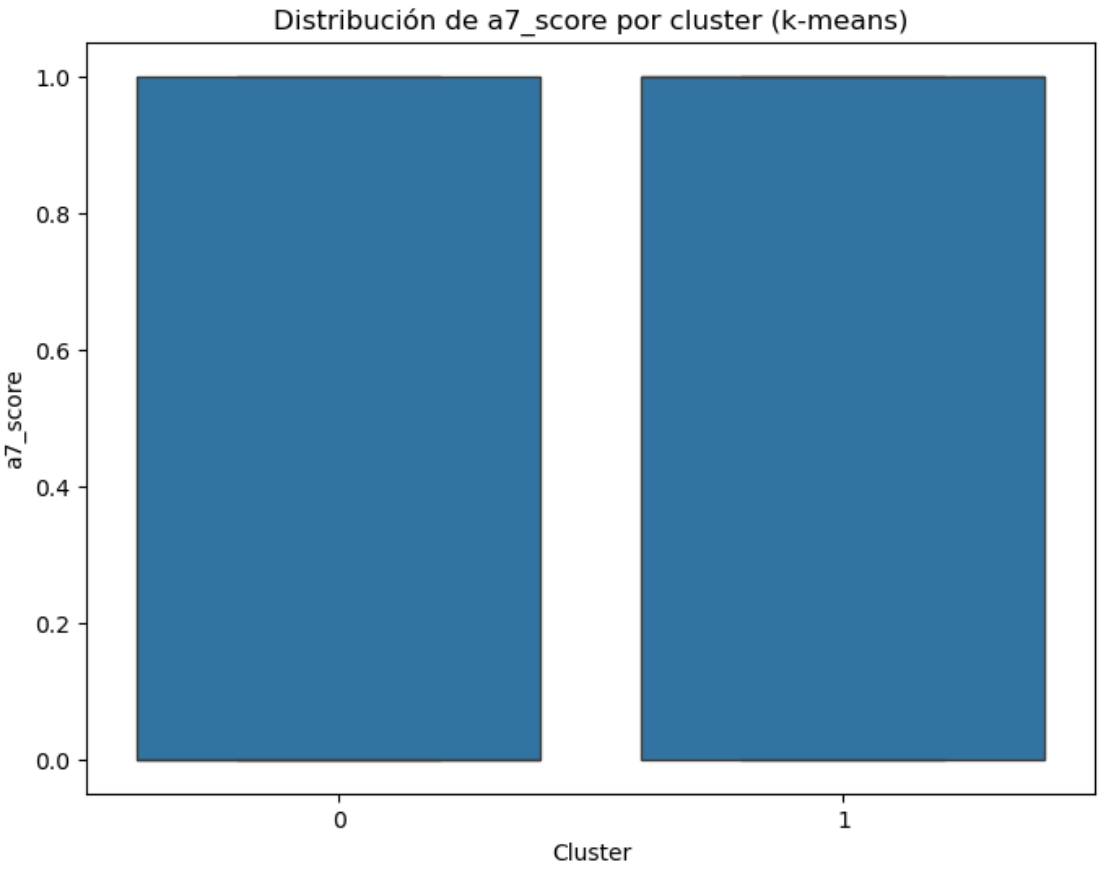

Distribución de a2_score por cluster (k-means)
a2_score
1.0
0.8
0.6
0.4
0.2
0.0
0
1
Cluster

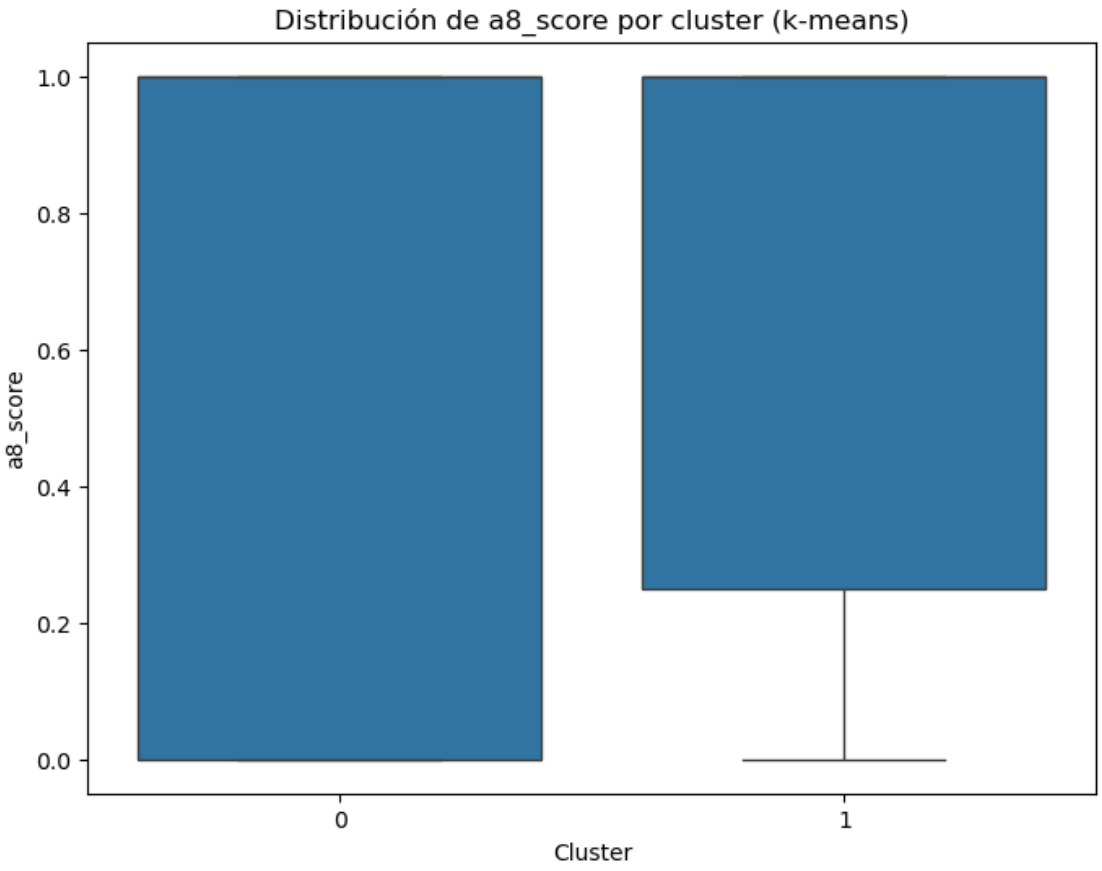

Distribución de a3_score por cluster (k-means)
a3_score
1.0
0.8
0.6
0.4
0.2
0.0
0
1
Cluster

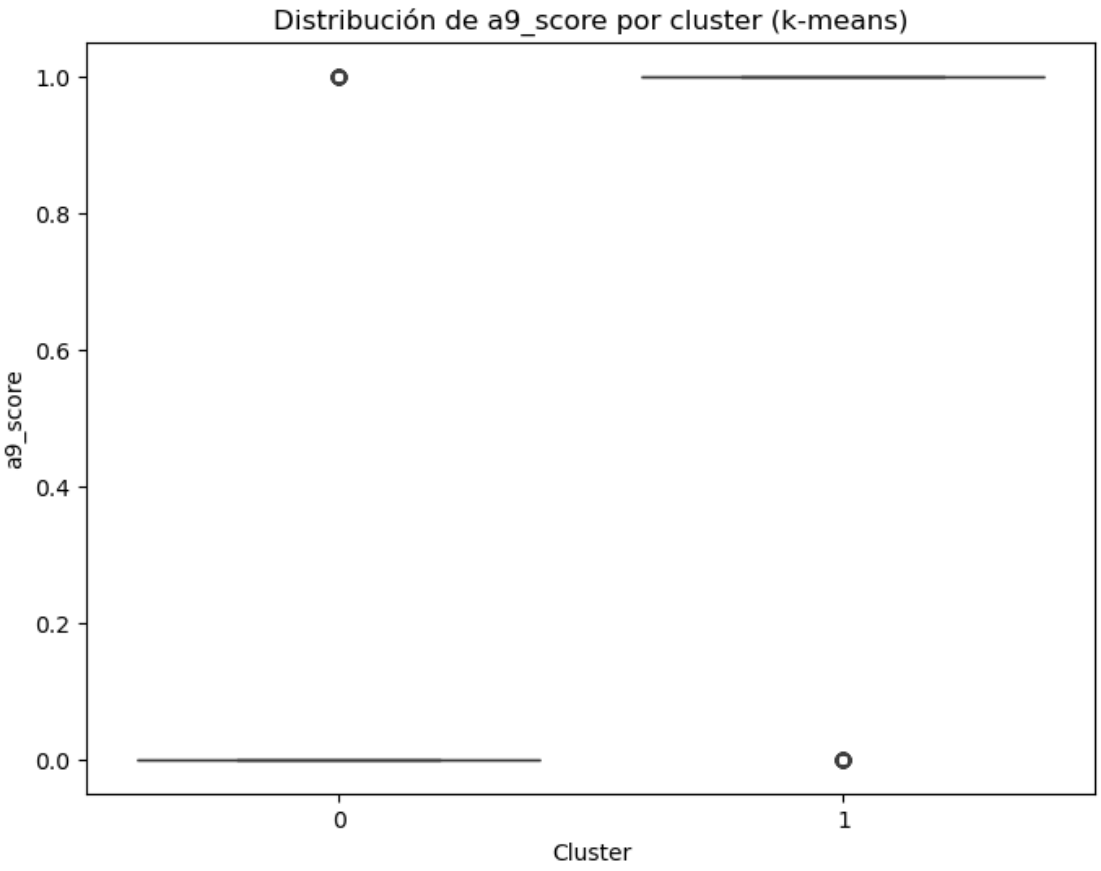

Distribución de a4_score por cluster (k-means)
a4_score
1.0
0.8
0.6
0.4
0.2
0.0
0
1
Cluster

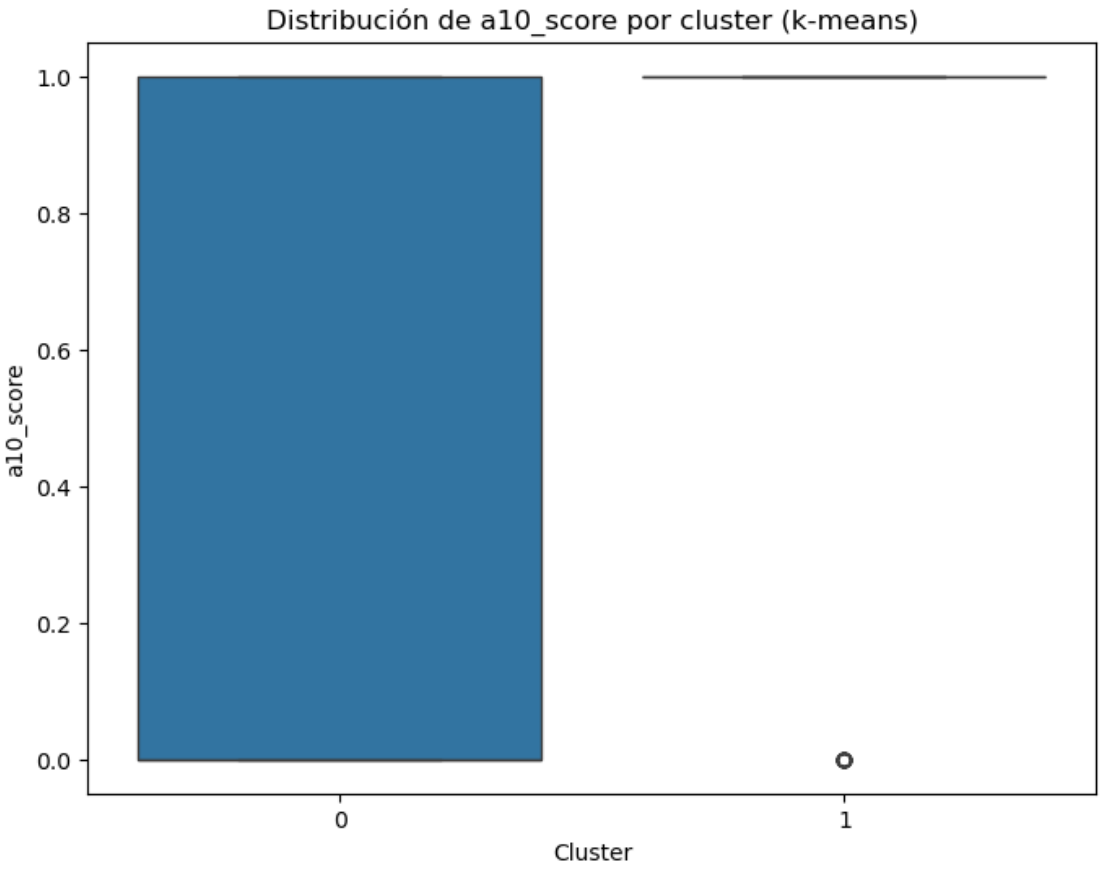

Distribución de a5_score por cluster (k-means)
a5_score
1.0
0.8
0.6
0.4
0.2
0.0
0
1
Cluster

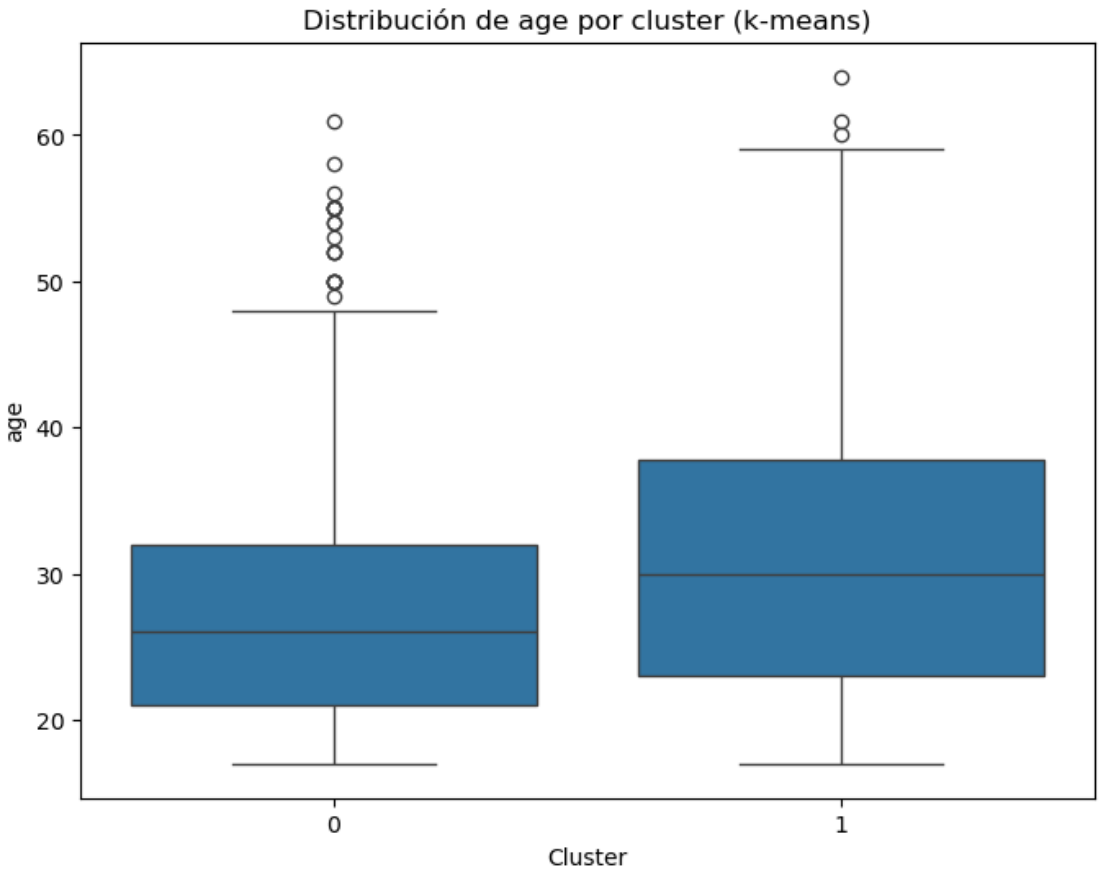

Distribución de a6_score por cluster (k-means)
a6_score
1.0
0.8
0.6
0.4
0.2
0.0
0
1
Cluster

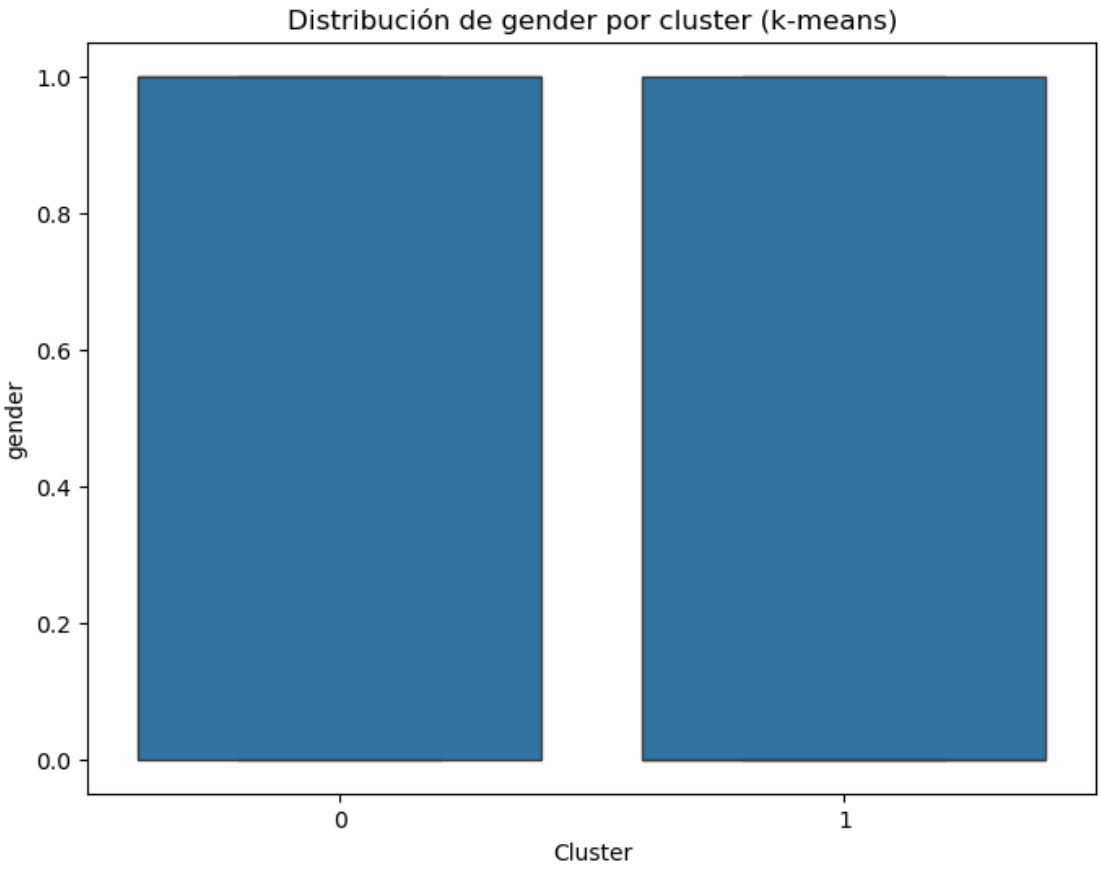

Distribución de a7_score por cluster (k-means)
1.0
0.8
0.6
0.4
0.2
0.0
a7_score
0
1
Cluster

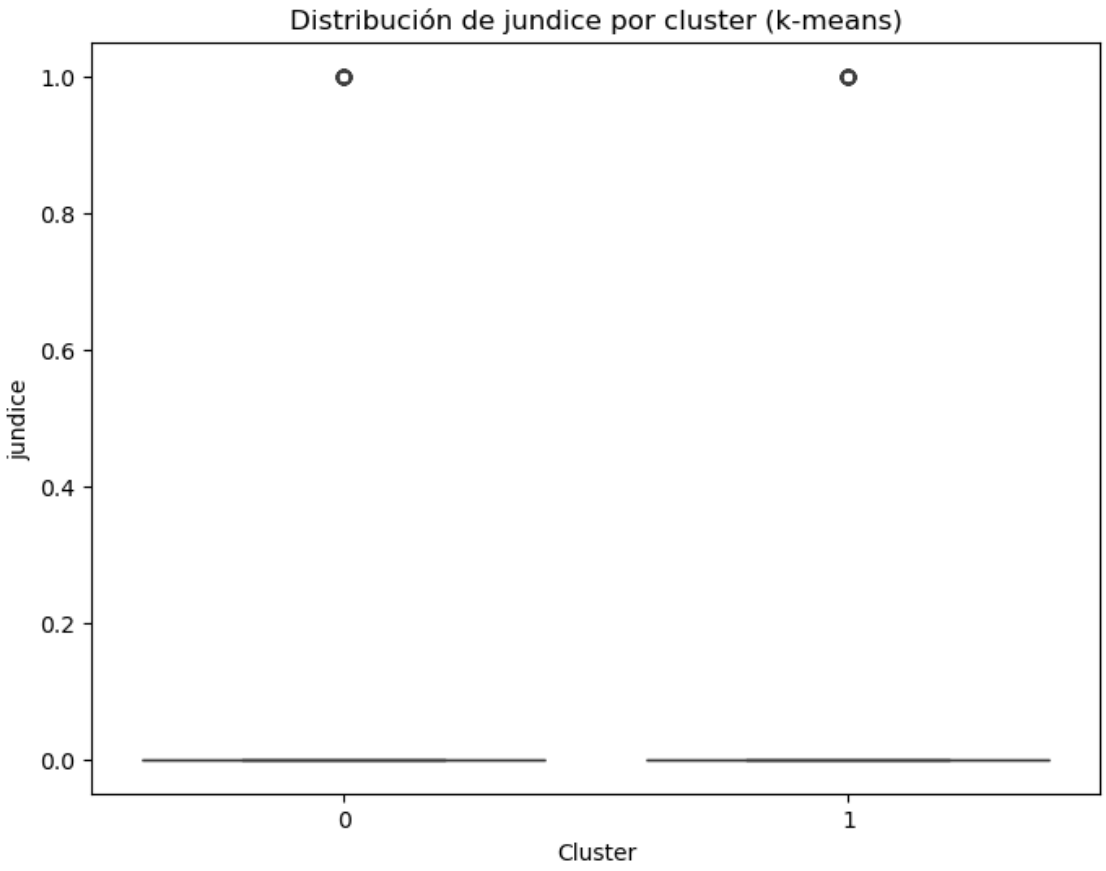

Distribución de a8_score por cluster (k-means)
a8_score
1.0
0.8
0.6
0.4
0.2
0.0
0
1
Cluster

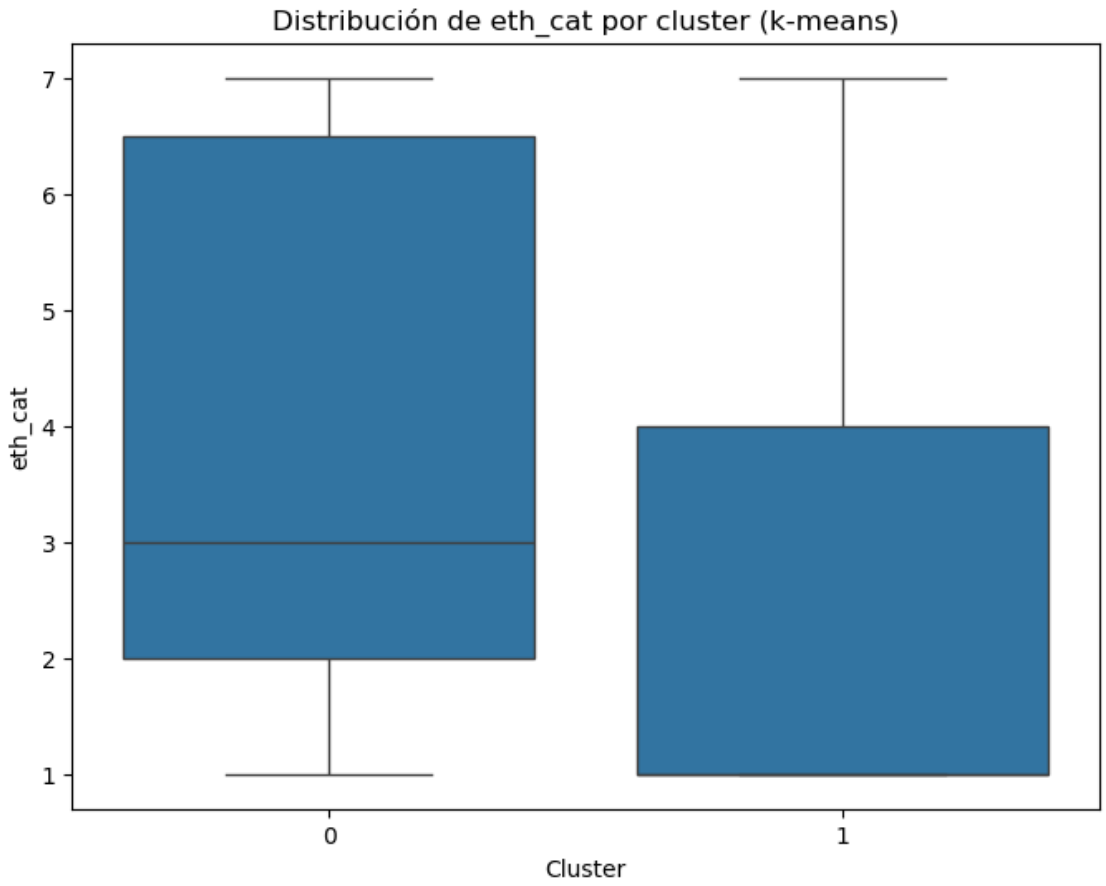

Distribución de a9_score por cluster (k-means)
1.0
0.8
0.6
0.4
0.2
0.0
a9_score
0
1
Cluster

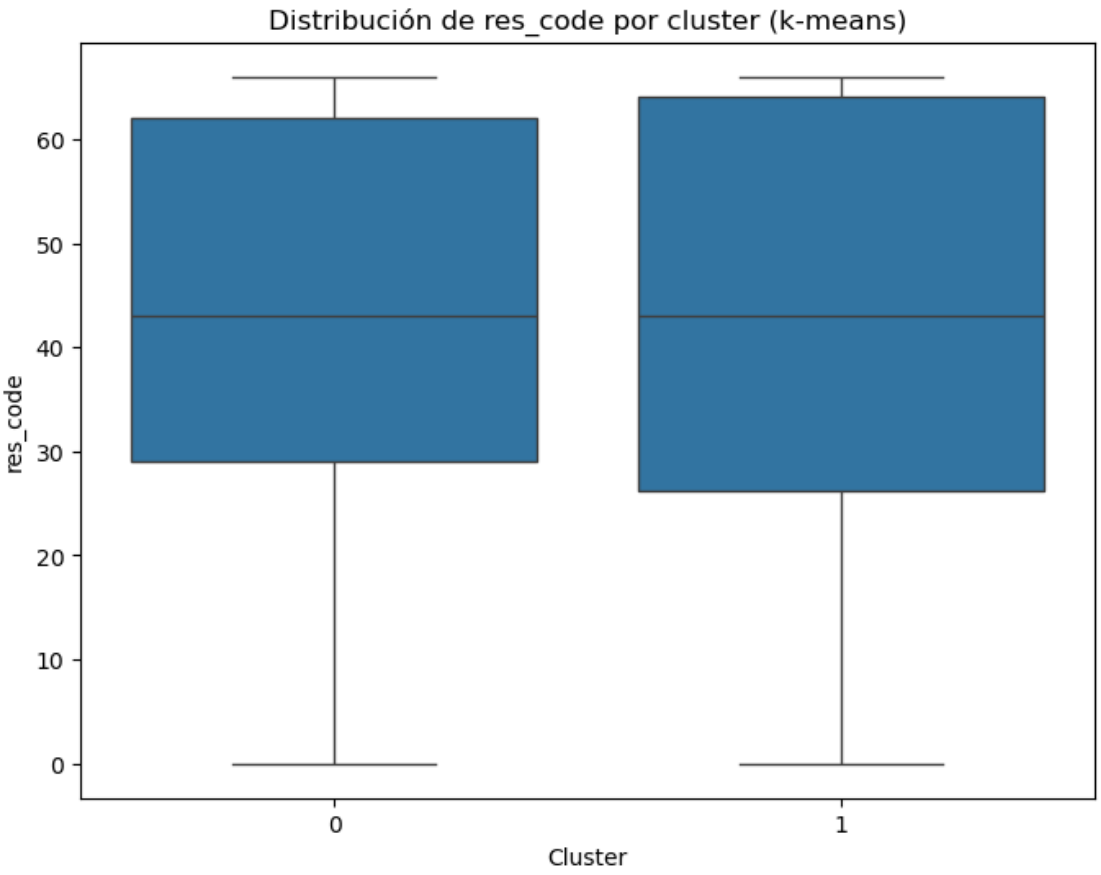

Distribución de a10_score por cluster (k-means)
a10_score
1.0
0.8
0.6
0.4
0.2
0.0
0
1
Cluster

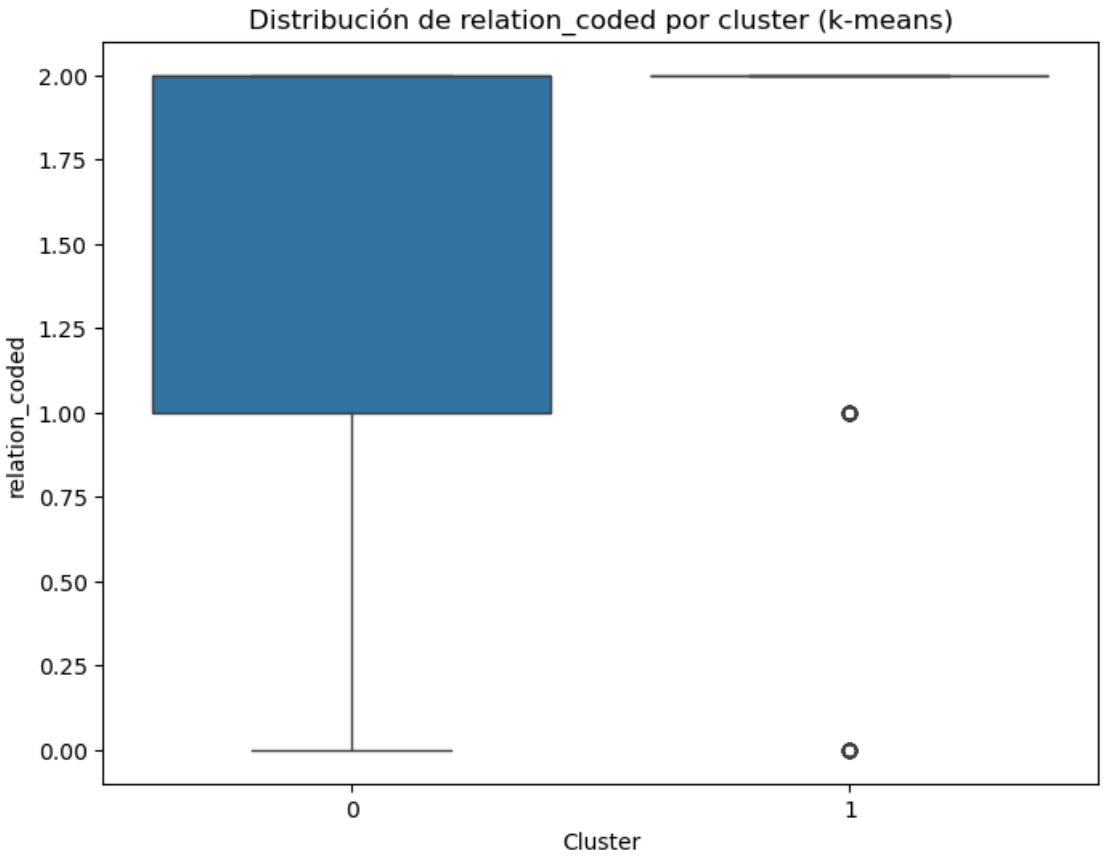

Distribución de age por cluster (k-means)
age
60
50
40
30
20
0
1
Cluster

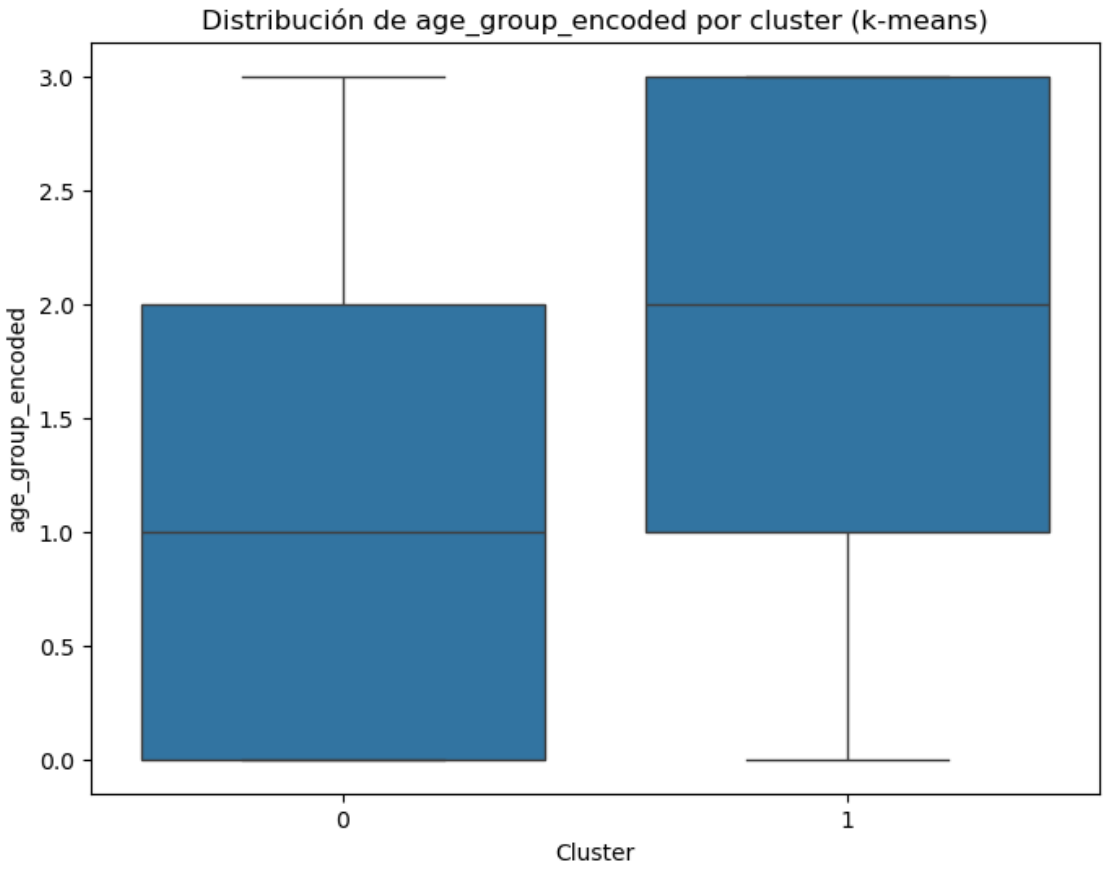

Distribución de gender por cluster (k-means)
gender
1.0
0.8
0.6
0.4
0.2
0.0
0
1
Cluster


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



Analysis of Feature Relevance in Cluster Formation
Exploratory Data Analysis (EDA) using the k-means clustering technique allowed for the evaluation of each feature's contribution to the formation of the two identified clusters.

AQ Scores
In particular, it was observed that the variables a7_score and a2_score exhibited a virtually identical distribution across both clusters, suggesting that these features did not provide discriminatory information to differentiate between the groups. The lack of variability in these variables between the clusters indicates that they were not associated with the underlying patterns that k-means used for clustering.

Gender
Similarly, the analysis of the gender variable revealed a similar distribution of males and females in both clusters. This suggests that gender, within the context of this dataset, does not appear to be a relevant factor in cluster formation or in the manifestation of Autism Spectrum Disorder (ASD).

Country of Residence
The analysis of the boxplot illustrating the distribution of res_code (country of residence code) by cluster in k-means revealed that the place of residence did not appear to be a determining factor in cluster formation.

- Similar distribution:

The medians and interquartile ranges of both clusters were found to be very similar, indicating that the distribution of country codes was practically the same in both groups.

- No discernible patterns in outliers:
  Although some outliers were observed in both clusters, no clear pattern was identified to suggest an association between specific countries and a particular cluster.

Given that the distribution of res_code was very similar in both clusters, it was concluded that the place of residence did not play a discriminatory role in the formation of the groups by k-means, suggesting that this variable does not provide significant information to distinguish between the two identified clusters.

JAUNDICE In the case of `jundice` we also conclude that jaundice does not necessarily seem to be related to Autism Spectrum Disorder. Most values are 0 in both clusters: The central lines of both boxes are at 0, meaning that most individuals in both clusters have a value of 0.

Age (Unscaled)
The variable `age` (undiscretized) appears to have some discriminatory ability between the two clusters generated by k-means.

- Higher median in Cluster 1: The central line inside the box (the median) is noticeably higher in Cluster 1 than in Cluster 0. This indicates that the typical age of individuals in Cluster 1 is higher than that of those in Cluster 0.
- Similar interquartile range: The height of the boxes (representing the interquartile range or IQR) is quite similar in both clusters, suggesting that the spread or variability of ages within each cluster is comparable.
- Outliers in Cluster 0: Cluster 0 shows several outliers (the individual points above the upper whisker), indicating that there are some individuals in this cluster with significantly older ages than most. Cluster 1, on the other hand, does not appear to have any outliers.

The difference in medians and the presence of outliers in Cluster 0 suggest that age, even without discretization, has some ability to separate the two clusters. This implies that age could be a factor that k-means has taken into account when forming the groups.

ETHNICITY Ethnicity (`eth_cat`) appears to be a factor influencing cluster formation, as there is a clear difference in the distribution of ethnic categories between the two groups.

- Cluster 0: Predominantly white.
- Cluster 1: More ethnically diverse, with a higher proportion of individuals from minority ethnicities or clustered in the "Other" category.
- Potential predictor: Ethnicity might be associated with genetic, environmental, or sociocultural factors that influence autism risk. Including it as a feature in models could improve predictive ability by capturing these differences.
- Identifying disparities: Analyzing the relationship between ethnicity and autism can help you identify potential disparities in the prevalence or diagnosis of the disorder between different ethnic groups. This can be valuable for developing more equitable intervention and support strategies.

Relationship Based on the coding for the variable `relation_coded`:

0: Parent 1: Relative 2: Self The analysis of the bar chart shows the distribution of relation_coded by cluster in k-means. We can conclude that:

Cluster 0: Contains a mix of individuals whose answers were provided by themselves ("Self") or by a relative ("Relative"). **Cluster 1:** Is composed almost exclusively of individuals whose answers were provided by themselves ("Self"). Therefore, we can infer that k-means has grouped the majority of individuals who came to the consultation alone at the time of diagnosis in Cluster 1.

Implications:

That the majority of individuals in Cluster 1 have come to the consultation alone could suggest that:

- Greater awareness or concern about autism: These individuals might have a greater awareness of their own traits or difficulties, leading them to actively seek an assessment.
- Greater autonomy: They might be more independent and proactive individuals in seeking information and support for their potential challenges.
- Less stigma or fear of diagnosis: Coming to the consultation alone could indicate less concern about the stigma associated with autism or a greater desire to obtain a clear diagnosis.

Other possible interpretations:

- Difficulties in social interaction: In some cases, people with autism may have difficulties interacting socially or communicating effectively, which could make them prefer to come to the consultation alone.
- Lack of family support: In other cases, a lack of support or understanding from family might lead individuals to seek help on their own.

Logistic factors: It is also possible that logistical factors, such as the availability of transportation or the need to care for other family members, may have influenced the decision to go to the clinic alone.

Importance of considering these implications:

It is essential to take these possible interpretations into account when analyzing the results of the model and when considering the inclusion of the variable `relation_coded` as a predictor characteristic.

- If the hypothesis of increased awareness or concern is true: This could mean that the model is learning to identify people who are more willing to seek help or who have greater knowledge about autism, rather than identifying autism itself.
- If the hypothesis of difficulties in social interaction is true: This could indicate that the model is capturing specific aspects of the manifestation of autism that relate to social interaction.
- If the lack of family support hypothesis is true: This could suggest that the model is indirectly identifying socioeconomic or family factors that may be associated with autism.

Age (Discretized) Age, even when discretized, appears to play a role in the clustering process. K-means has partially separated individuals based on their age group, with Cluster 0 tending towards younger individuals and Cluster 1 towards older individuals.

- Not the only factor: The overlap in the age distribution between clusters indicates that factors other than age are also contributing to the formation of the clusters.

Possible implications:

- Age-related patterns in autism: This could suggest that there are age-related patterns or differences in the characteristics associated with autism within your dataset.

Keeping features that do not contribute information to the distinction between clusters can introduce noise into the model and potentially affect its performance, so we proceeded to eliminate those features that we did not need:

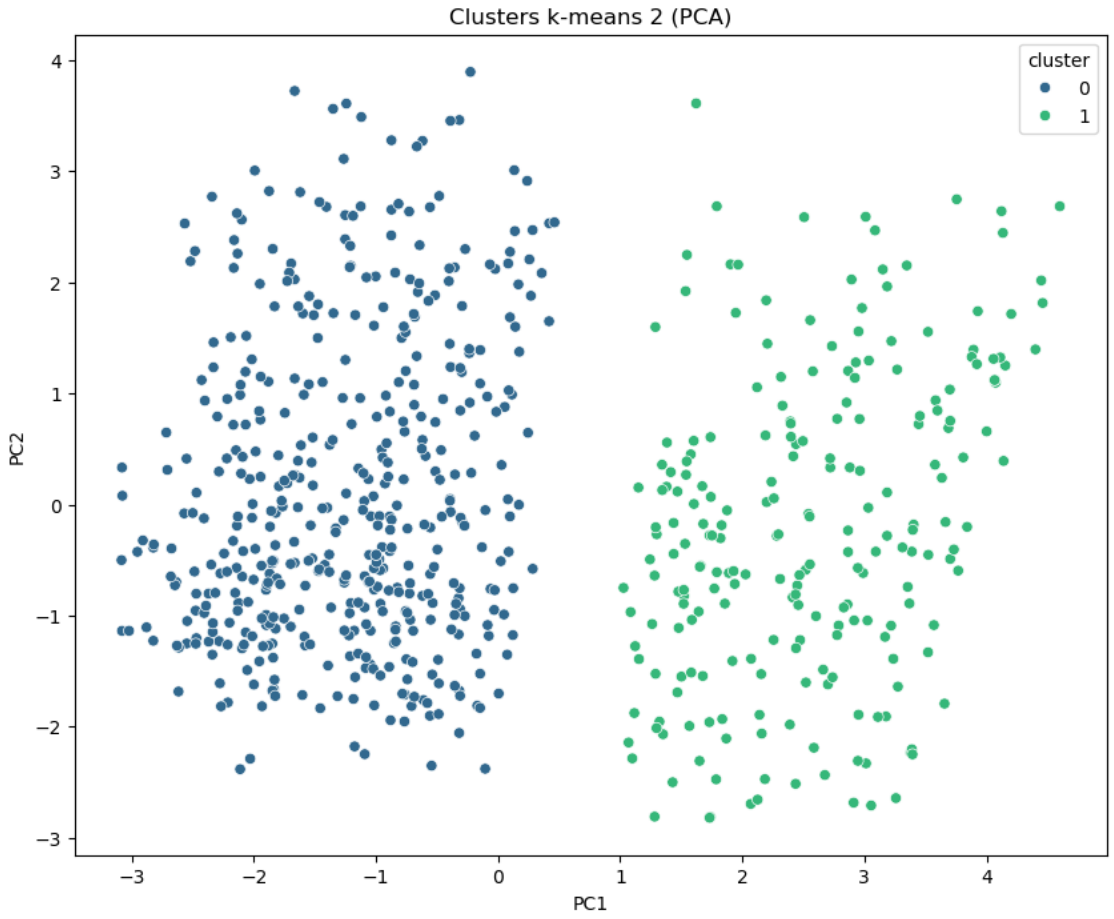
[ ]:
```
# eliminamos columnas
df.drop(columns=['a7_score', 'a2_score', 'gender', 'res_code', 'jundice'],
        inplace=True)
# Revisamos el dataframe
df.info()
```

We applied K-Means again and observed the resulting clustering after applying modifications to the dataset:

[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)


# 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
```



We observed that it continued to group almost identically after eliminating the variables that do not contribute to grouping the cases, so we can eliminate the redundant feature:

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

DBSCAN and OPTICS Models Density-based unsupervised clustering models, such as DBSCAN (Density-Based Spatial Clustering of Applications with Noise) and OPTICS (Ordering Points To Identify the Clustering Structure), were considered as potential alternatives for data analysis. These models, unlike k-means which is centroid-based and assumes spherical cluster shapes, iden-

tify groups based on the density of points in the feature space, allowing them to discover clusters of arbitrary shapes and handle noisy data more effectively.

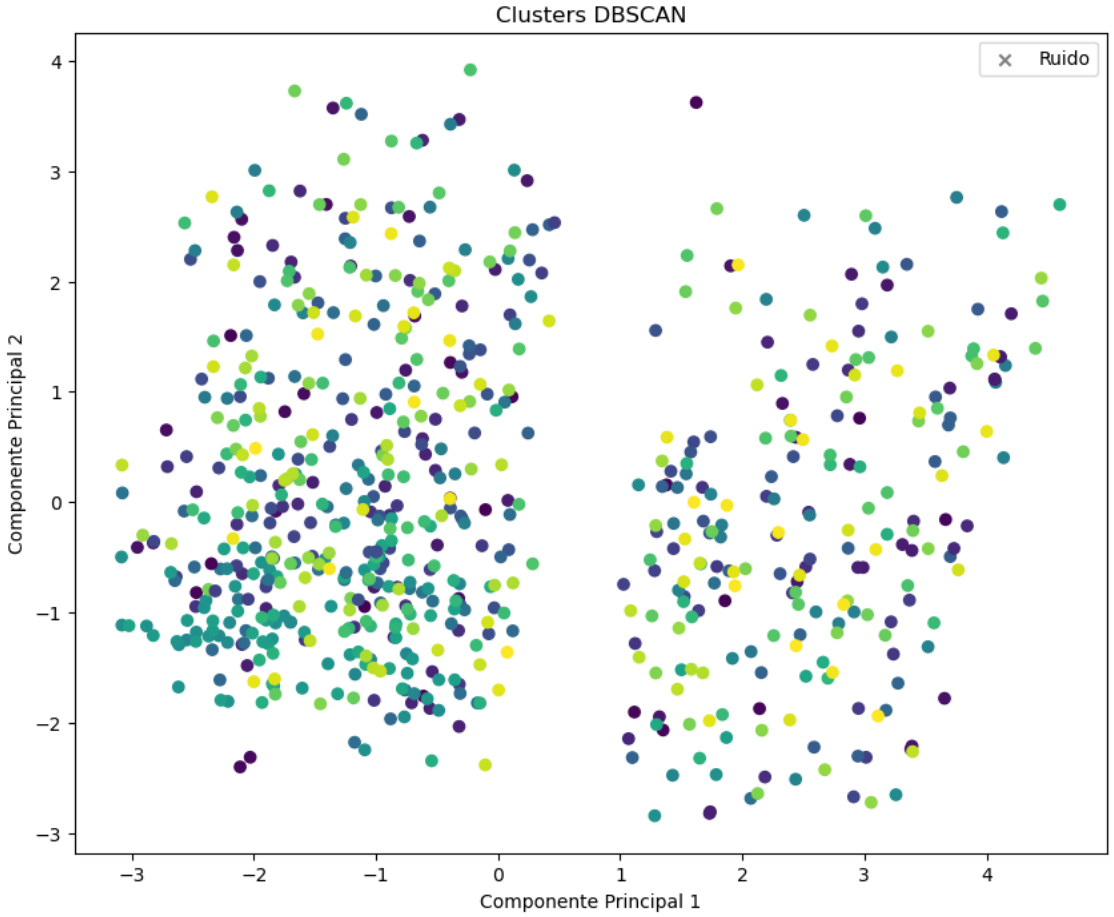
[71]:
```
from sklearn.cluster import DBSCAN

# 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
```

We applied PCA to reduce the dimensionality to 2 components, although according to the results of K-Means, we know that it tends to group into two clusters:

[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')
```

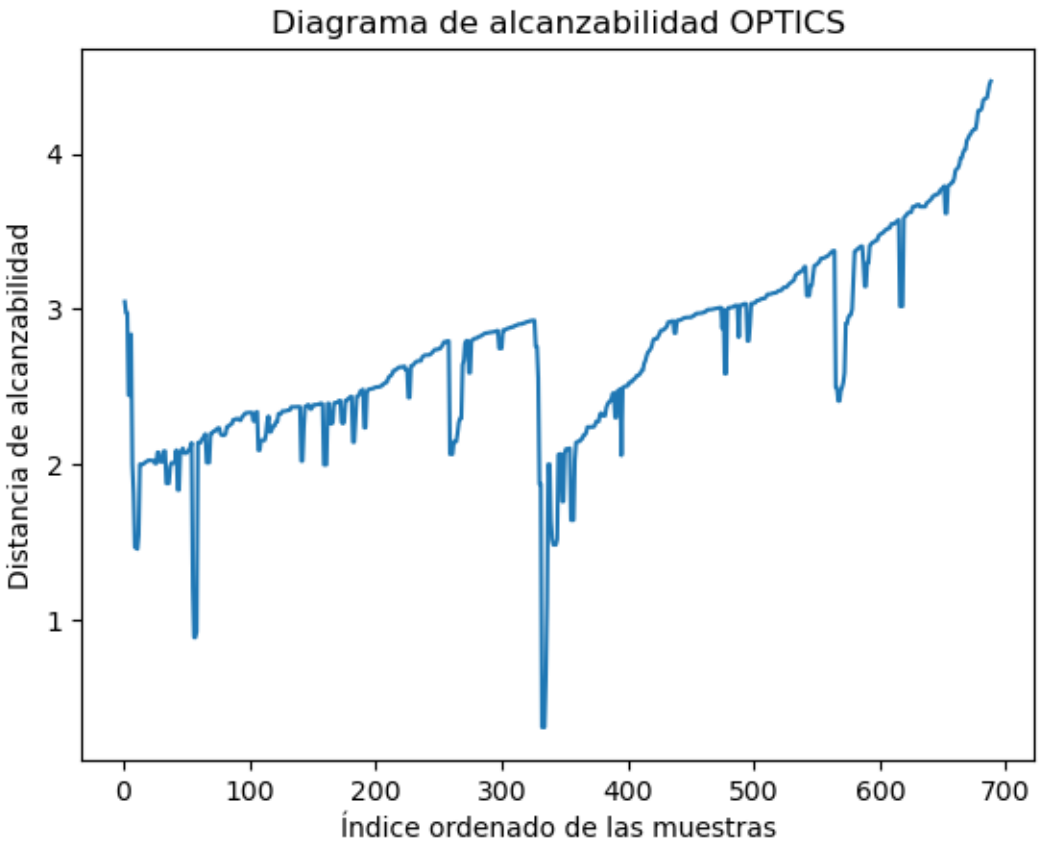
```
plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.title('Clusters DBSCAN')
plt.legend()
plt.show()
```



The scatter plot suggested that *DBSCAN* has not found a clear cluster structure in the data after setting ***different hyperparameters***. We tried with *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()
```



OPTICS Reachability Diagram Analysis The reachability diagram generated by OPTICS reveals crucial information about the underlying density structure in the data. In this graph, each point represents a sample and its reachability distance, which indicates the proximity of a point to other points in the data set, considering the density of its neighborhood.

Key Observations

- Evidence of clustering: Despite a possible first impression of a single dominant cluster, several steep valleys are observed, particularly in the first half of the graph. These valleys suggest the existence of multiple clusters with relatively high densities, separated by regions of lower density (the peaks).
- Dominant cluster and possible noise: Towards the end of the graph, the reachability distance increases considerably and the curve smooths out, with fewer steep valleys. This could indicate

the presence of a large, less dense cluster, or the existence of noise or outliers.

- Potential outliers: Isolated peaks rising above valleys could represent outliers, i.e. points that are relatively far from other points and do not belong to any dense cluster.

Possible Conclusions

- Cluster structure: The data appears to present a clustering structure, with several smaller, denser clusters in the first part of the graph, and a larger, less dense cluster or noise in the final part.
- Estimation of the number of clusters: The exact number of clusters is difficult to determine visually, but it is estimated between 3 and 5 main clusters based on the most pronounced valleys. The final choice will depend on the objectives of the analysis and the context of the problem.
- Presence of noise: Isolated peaks in the graph suggest the possible presence of noise or outliers in the data.

We will extract the clusters:

[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
```

A silhouette score of 0.040 is very low, indicating that the clusters obtained with *OPTICS* are

not well defined and there is probably significant overlap between them. Considering that *k-means* managed to identify two clusters with a clearer separation, it seems reasonable to conclude that *OPTICS* is not the most suitable algorithm for this particular dataset.

We removed the columns from `dbscan` and `optics` whose results were unsatisfactory and may confuse future models:

[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
```

Exploratory data analysis (EDA) and data quality assurance (DQA) (II) ***Deepening the exploration of features***

After a first phase of EDA and DQA, and having obtained new knowledge about the structure

of the data and the relationships between variables, we proceed to a second phase of exploratory analysis. In this stage, we will focus on deepening the understanding of individual characteristics and their possible relevance for the prediction of autism.

Age as a factor in clustering

Based only on the boxplot we obtained above, we can say that age has some discriminatory power in separating the two clusters, but it is not a very strong classifier. The slight difference in the medians and the presence of outliers in Cluster 0 indicate that age might play a role in clustering, but it is likely not the only or dominant factor.

- Difference in median age:

The median age (represented by the horizontal line inside each box) is slightly higher in Cluster 1 compared to Cluster 0. This suggests that, on average, **individuals in Cluster 1 tend to be older than those in Cluster 0**.

- Interquartile range (IQR):

The IQR (the height of each box) is visually similar between the two clusters, indicating that the spread or variability of ages within each cluster is roughly comparable.

- Outliers: Cluster 0 exhibits some outliers (the individual points beyond the whiskers), representing individuals with significantly higher ages than the rest of the cluster. Cluster 1, on the other hand, appears to have no outliers. This suggests that Cluster 0 has a wider age range, while Cluster 1 is more homogeneous in terms of age.

Ethnicity as a factor in clustering: In the same vein as age, ethnicity appears to be a factor influencing cluster formation, as there is a clear difference in the distribution of ethnic categories between the two groups.

- Cluster 0: Predominantly white.
- Cluster 1: More ethnically diverse, with a higher proportion of individuals from minority ethnicities or clustered in the "Other" category.

Ethnicity can help in supervised models as:

- Potential predictor: Ethnicity might be associated with genetic, environmental, or sociocultural factors that influence autism risk. Including it as a feature in models could improve their predictive ability by capturing these differences.
- Identification of disparities: Analyzing the relationship between ethnicity and autism can help you identify potential disparities in the prevalence or diagnosis of the disorder between different ethnic groups. This can be valuable for developing more equitable intervention and support strategies.

Deepening the relationship between ethnicity and autism

1. Contingency table:

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

```
Valor p: 0.00000075876952605960
```

Contingency table analysis:

The contingency table and associated p-value indicate a statistically significant association between ethnicity (eth_cat) and autism diagnosis (austim).

- p-value: The p-value is extremely small (7.58e-07), well below the typical significance level of 0.05. This indicates that it is **highly unlikely that the observed association between ethnicity and autism is due to chance**.
- Interpretation: We can conclude that there is sufficient evidence to **reject the null hypothesis** that there is no association between ethnicity and autism in this sample. Ethnicity appears to be related to autism diagnosis in this data set.

- Magnitude of association: Cramer's V coefficient

The Cramer's V coefficient, which measures the strength of the association between categorical variables, was calculated to be 0.238. This value suggests a low to moderate association between ethnicity and autism in this sample.

Limitations:

While the chi-square test and Cramer's V coefficient tell us about the existence and magnitude of an association, they do not reveal the nature or direction of this relationship. For a deeper understanding, it is necessary to carefully examine the contingency table and consider other factors that may be influencing this association.

Recommendations for future analysis:

- Residual analysis: Examining the standardized residuals of the contingency table can help identify which cells contribute most to the statistical significance and, therefore, to the observed association.
- Control for confounding variables: It is important to consider the possibility that other factors, such as socioeconomic status or access to health services, may be influencing both ethnicity and autism diagnosis, generating a spurious association. Additional analyses that control for these variables are recommended to assess whether the association between ethnicity and autism persists.
- Longitudinal studies: To establish a possible causal relationship, longitudinal studies would be needed that follow individuals over time and assess how ethnicity and other factors interact in the development and diagnosis of autism.

It is critical to remember that statistical association **does not imply causation**. Ethnicity may be a marker for other underlying factors that are actually related to autism.

2. Comparison of proportions: **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   |
```

Analysis of proportion comparison: The image shows two tables that provide information about the distribution of autism diagnoses across different ethnic categories within your dataset.

1. Table 2: "Autism Percentage by Ethnicity"

This table shows the percentage of individuals within each ethnic category who have been diagnosed with autism.

- Key Observations: The "Latino" category has the highest percentage of autism diagnoses (29.03%), suggesting a possible overrepresentation of autism within this group in the sample. The "Asian" category has the lowest percentage (3.28%), indicating a possible underrepresentation. Other categories show varying percentages, with "Other" at 7.97% and "Black" at 11.90%.

2. Table 3: "Percentage of Total Diagnosed by Ethnicity"

This table presents the percentage contribution of each ethnic category to the total number of autism diagnoses in the data set.

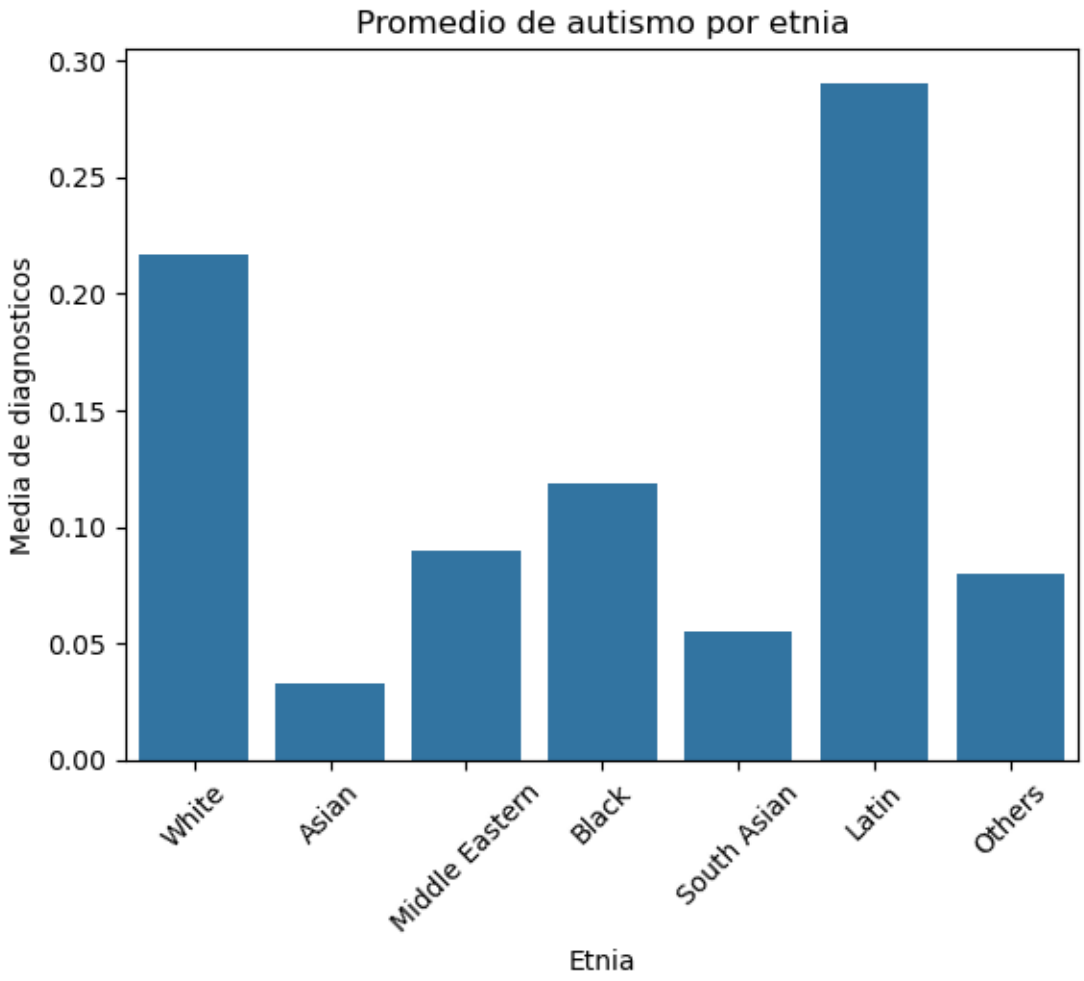
- Key Observations:
- The "White" category contributes the most to the total number of autism diagnoses (56.18%), which is expected given that it is the largest group in your sample.
- Despite having a high percentage of autism diagnoses within its own category, the "Latino" group contributes only 3.03% to the total number of diagnoses, likely due to its smaller sample size.
- The "Other" category contributes 12.36% to the total, suggesting that although the percentage within the group is moderate, it still represents a significant portion of diagnosed individuals.

3. Visualizations:
   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)
```

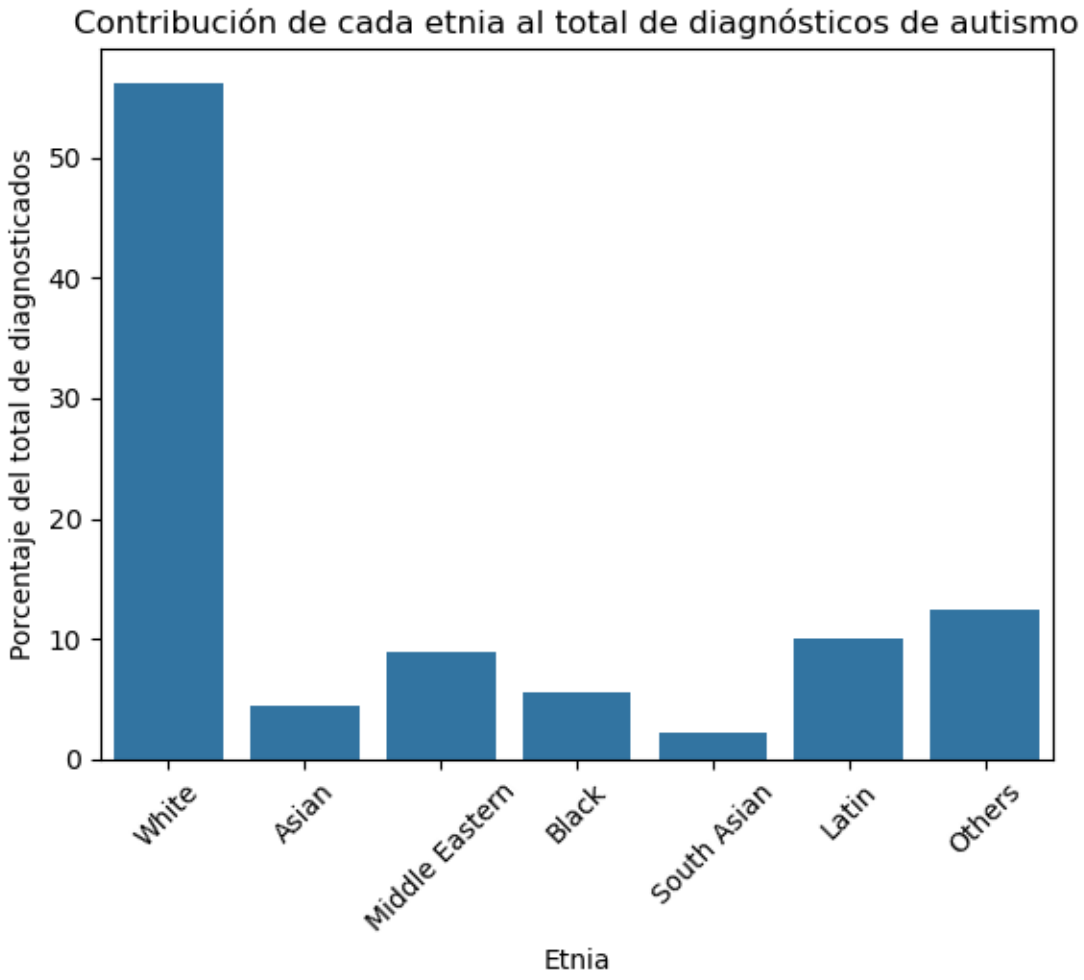
```
{1: 'White', 2: 'Asian', 3: 'Middle Eastern', 4: 'Black', 5: 'South Asian', 6:
'Latin', 7: 'Others'}
```

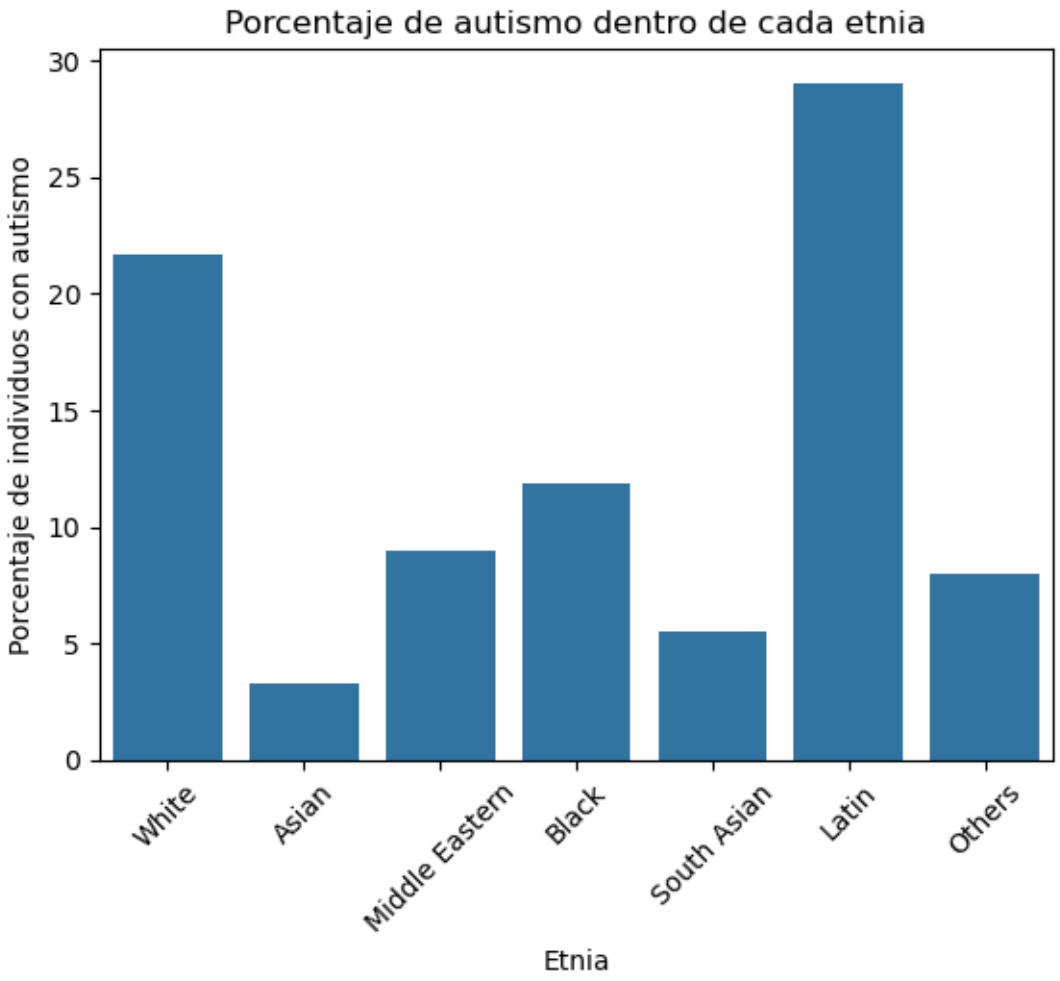
[69]:
```
import seaborn as sns

# 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. Logistic regression analysis: ***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
```

Interpretation of the results of the linear logistic regression for ethnicity:

- Ethnicity coefficient: -0.1217. This negative coefficient suggests that, in general, as the numerical code assigned to ethnicity increases, the probability of being diagnosed with autism decreases.
- p-value: 0.00000027074108759248. This p-value is lower than the typical significance level of 0.05, indicating that the association between ethnicity and autism is statistically significant. **That is, ethnicity appears to be a significant predictor of autism in the model.**
- ***Analysis of the relationship between ethnicity and autism diagnosis*** Exploratory data analysis (EDA) has allowed us to examine the possible association between ethnicity and the diagnosis of Autism Spectrum Disorder (ASD) in the sample studied. Two complementary approaches were used to assess this relationship:

1. Proportion of autism by ethnicity:

The percentage of individuals diagnosed with autism within each ethnic category was calculated. The results revealed significant variability in these proportions, with the "Latino" category presenting the highest percentage (29.03%), followed by the "Black" category (11.90%). In contrast, the "Asian" category showed the lowest proportion (3.28%).

2. Contribution of each ethnicity to the total number of diagnoses:

The percentage contribution of each ethnic category to the total number of autism diagnoses in the sample was analyzed. The "White" category contributed the most (56.18%), which is expected given that it is the largest group in the sample. Despite having a high proportion of autism within its own category, the "Latino" group contributed only 3.03% to the total number of diagnoses, likely due to its smaller sample size.

Chi-square test and Cramer's V coefficient:

A chi-square test was performed to assess the statistical significance of the association between ethnicity and autism. The p-value obtained (7.58e-07) was significantly lower than the typical significance level of 0.05, indicating a statistically significant association between both variables.

Interpretation:

The results suggest a possible relationship between ethnicity and autism diagnosis in the sample studied. However, it is important to interpret these findings with caution, taking into account the following considerations:

- Correlation vs. Causality: The observed statistical association does not necessarily imply a direct causal relationship between ethnicity and autism. There could be other factors, such as socioeconomic or cultural differences, that influence both ethnicity and diagnosis.
- Sample size and representativeness: Some ethnic categories have a small number of representatives in the sample, which could affect the reliability of the estimates. Furthermore, the sample might not be perfectly representative of the general population, limiting the generalizability of the results.
- Possible biases: It is crucial to consider the possibility of biases in data collection or diagnostic practices that may have influenced the results. Conclusions:

The exploratory analysis of the relationship between ethnicity and autism suggests a statistically significant association. Future research with larger, more representative samples is required to confirm these findings and explore the factors underlying this association. It is crucial to consider possible limitations and biases when interpreting these results and to avoid generalizations or stereotypes based on ethnicity.

AGE_GROUP_ENCODED

Going deeper into the relationship between age and autism

1. Contingency table:

[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
```

Analysis of the Relationship between Age and Autism Diagnosis

Results of exploratory analysis focused on the relationship between age, discretized in quartiles (age_group_encoded), and autism diagnosis (austim). Two main approaches were used: the chi-square test and logistic regression.

- Chi-square Test and Cramer's V Coefficient The chi-square test yielded a p-value of 7.33e-07, significantly lower than the threshold of 0.05. This indicates a statistically significant association between discretized age and autism diagnosis in the sample. The Cramer's V coefficient, which quantifies the strength of this association, was 0.213, suggesting a relationship of low to moderate magnitude.
- Logistic Regression Analysis A coefficient for the variable age_group_encoded of -0.5689 with a p-value of 2.71e-07. This negative and statistically significant coefficient supports the conclusion of the chi-square test, indicating that as the age quartile increases, the probability of an autism diagnosis decreases.
- Interpretation of Results These findings suggest a trend in the sample where younger individuals have a higher probability of being diagnosed with ASD compared to older individuals. This observation could be related to increased awareness and early detection of autism in recent decades, changes in diagnostic criteria, or the greater ease of identifying certain traits associated with autism at early ages.
- Limitations and Considerations It is important to recognize that this association does not imply causality. Other factors, such as socioeconomic or cultural differences, could be influ-

encing both age at diagnosis and the presence of ASD. In addition, generalization of these results to the general population should be done with caution due to possible sample bias and variability in diagnostic practices.

- Conclusions The analysis presented shows a statistically significant association between age and autism diagnosis in the sample studied. The results suggest that younger individuals tend to have a higher probability of being diagnosed. However, further research is required to understand the underlying causes of this association and to assess its clinical relevance. It is essential to consider other factors and possible biases when interpreting these findings.

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'))
```

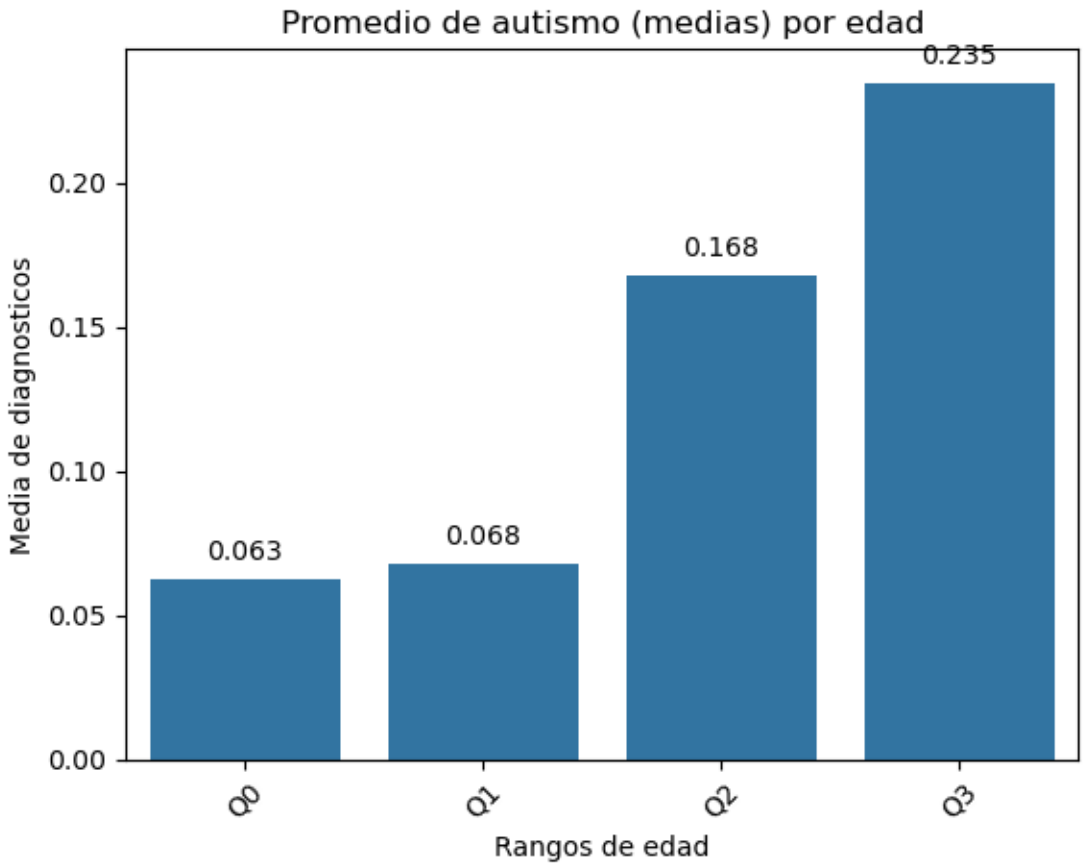
```
Conteo de individuos por edad:
 | age_group_encoded   | count   |
|:--------------------|:--------|
| 0                   | 175     |
| 1                   | 191     |
| 2                   | 161     |
| 3                   | 162     |

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

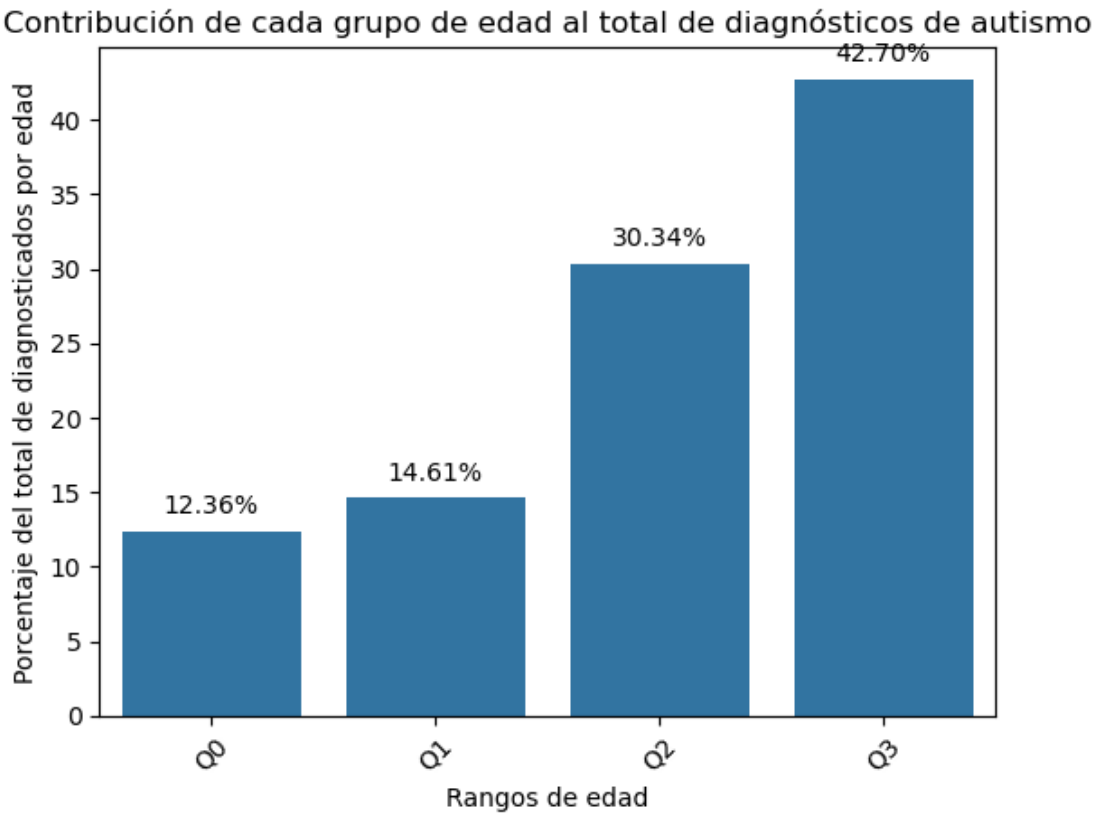
[76]:
```
# Creamos un diccionario con las etiquetas de
# 'age_group_encoded':
age_group_map = {
    0 : "Q0",
```

```
    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
```

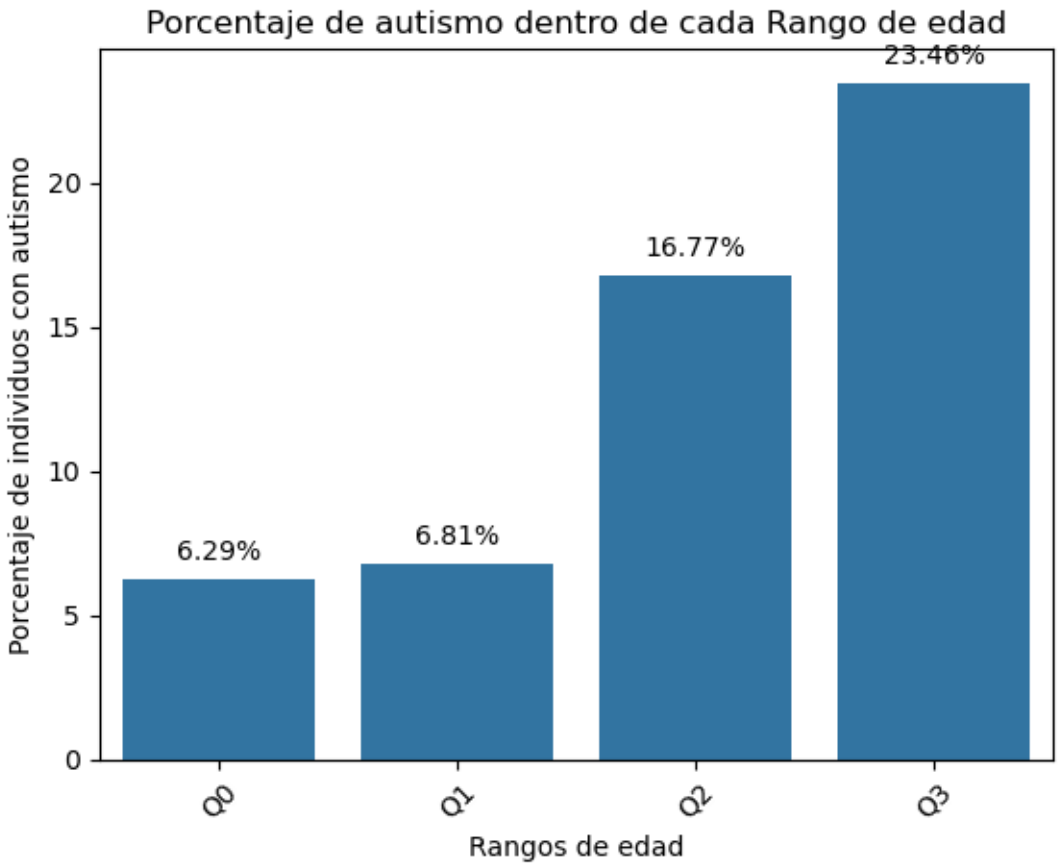
[76]: <function matplotlib.pyplot.show(close=None, block=None)>



[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. Logistic regression analysis: ***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
```

Analysis of the results of the logistic regression model for discretized age The logistic regression analysis, using age discretized into quartiles as a predictor of the diagnosis of Autism Spectrum Disorder (ASD), has revealed a statistically significant association between both variables.

The coefficient obtained for discretized age was 0.5689, with a p-value of 2.71e-07. This positive coefficient indicates that, in general, as the age quartile increases (from Q1, the youngest, to Q4, the oldest), the logarithm of the odds (log-odds) of being diagnosed with ASD increases.

The p-value, being significantly lower than the conventional significance level of 0.05, supports the conclusion that age, even in its discretized form, is a relevant predictor of autism in this sample.

Other statistics:

- Pseudo R-squared: 0.0548 This metric, also known as McFadden's R-squared, measures the goodness of fit of the logistic regression model. It ranges from 0 to 1, with higher values indicating better fit. In this case, the value of 0.055 suggests that the model explains only a small portion of the variance in autism diagnosis.
- Log-likelihood: -250.607 This is the logarithm of the likelihood function of the fitted model. It is a measure of how well the model fits the observed data. Higher values indicate better fit.
- AIC: 505.215, BIC: 514.285 AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion) are metrics used for model selection. They penalize models with more

parameters to prevent overfitting. Lower AIC and BIC values indicate a better model fit, considering its complexity.

Interpretation and considerations:

These results suggest that, in this sample, individuals in the older age groups tend to have a higher probability of being diagnosed with ASD compared to younger individuals. This observation could be related to several factors, such as:

- Greater difficulty of diagnosis in adults: Autism can manifest differently in adults, which could make it difficult to identify and diagnose compared to children, leading to later diagnoses.
- Bias in the sample: It is also possible that the sample is biased towards older individuals with autism.

1.5 DATASET DESCRIPTION

The dataset resulting from the ETL process contains information on 689 adults who participated in a screening study for Autism Spectrum Disorder (ASD). Each record includes the following characteristics:

1. **AQ-10 test scores** *(a(x)_score)*: *Ten questions with binary answers (yes/no) designed to assess autistic traits. Values 0-1.* **items `a2_score` and `a7_score` have been removed***
2. **Age** *(age): Age of the participant in years.
3. **Gender** *(gender)*: Coded as 0 (female) or 1 (male).***Removed in the final version for supervised models***
4. **Jaundice at birth** *(jundex)*: Coded as 0 (no) or 1 (yes). ***Removed in final version for supervised models***
5. **Autism diagnosis** *(austim)*: Coded as 0 (no) or 1 (yes).
6. **Total score on the AQ-10** *(result)*: Sum of the scores on the ten questions. Range 0-10.
7. **Binary result of the AQ-10 result** *(aq_binary)*: Binary for diagnoses with

$$result >= 6$$

8. **Ethnicity** *(eth_cat)*: Ethnic category of the participant, coded numerically. {'White': 1, 'Asian': 2, 'Middle Eastern': 3, 'Black': 4, 'South Asian': 5, 'Latin': 6, 'Others': 7}
9. **Country of residence** *(res_code)*: Numerically coded.
10. **Relationship to the person who provided the information** *(relation_coded)*: Numerically coded {"Parent": 0, "Relative": 1, "Self": 2}. Removed fron the final dataset.

This dataset was cleaned and preprocessed to remove missing values and inconsistencies, and to encode the categorical variables in a format suitable for use in machine learning models.

1.5.1 CLOSING THE ETL SECTION

With the completion of the ETL process, we have managed to transform the original dataset ***"Autism Screening Adult"*** into a format suitable for analysis and modeling. Data quality issues such as missing values and inconsistent categories have been addressed, and categorical variables have been coded to facilitate their use in machine learning algorithms.

The resulting dataset provides a solid foundation for the next stage of the project, in which we will explore different models for predicting ASD risk in adults. We hope that this dataset will

contribute to a better understanding of the factors that influence autism and to the development of more accurate tools for its early detection.

[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 UNSUPERVISED MODELS

1.6.1 Logistic Regression Models - Binary Classification (I)

In this section, we will build a binary classification model to predict the diagnosis of Autism Spectrum Disorder (ASD) in adults. We will use the target variable austim, which is already coded as 0 (non-autistic) and 1 (autistic).

Our goal is to train a machine learning algorithm that can learn to distinguish between these two

classes from the features available in the dataset, such as responses to the AQ-10 questionnaire, age, ethnicity, and other relevant factors.

We will evaluate the performance of the model using appropriate metrics for binary classification, such as precision, recall, F1-score, and area under the ROC curve (AUC-ROC). These metrics will allow us to determine the model's ability to correctly identify individuals with and without ASD, as well as its ability to discriminate between both classes.

Through this model, we seek to develop a tool that can help health professionals identify people at higher risk of ASD and facilitate access to early diagnosis and treatment.

Aspects to consider: - Feature selection: We will explore different combinations of features to determine which are most relevant to predict autism. - Classification algorithms: We will experiment with various classification algorithms, such as logistic regression, decision trees, and support vector machines, to identify the one that best fits our data. - Hyperparameter optimization: We will adjust the hyperparameters of the models to maximize their performance. - Interpretability: We will look for models that are not only accurate, but also interpretable, to understand which factors are most important in predicting autism.

Evaluation metrics: - Accuracy: Proportion of correct predictions out of the total predictions. - Recall (Sensitivity): Proportion of positive cases (autism) correctly identified. - F1-score: Harmonic mean between precision and recall, which balances both metrics. - AUC-ROC: Area under the ROC curve, which measures the model's ability to distinguish between positive and negative classes.

Objectives: - Develop an accurate and robust binary classification model to predict ASD diagnosis in adults. - Identify the most relevant features for the prediction of autism. - Evaluate the performance of the model using appropriate metrics. - Contribute to the understanding of the factors that influence the risk of ASD.

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

We are going to create stratified splits for training and testing, since as we have elucidated previously, there is a class imbalance.

[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
```

Dense Neural Network (DNN) or Multilayer Perceptron (MLP) model. Artificial neural network where the layers are stacked sequentially one after the other. In this particular case, a `sequential` *Keras* model that has multiple dense layers (fully connected) with non-linear activation functions.

We are going to create a Dense Neural Network model using a fit of the SMOTE samples and employing 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}
```

Analysis and Conclusions of the Neural Network Model with SMOTE and One-Hot Encoding The implemented neural network model, using the SMOTE oversampling technique and One-Hot Encoding for categorical variables, has shown moderate performance in the binary classification task of Autism Spectrum Disorder (ASD) in adults. Despite hyperparameter optimization efforts using Keras Tuner, the evaluation metrics reveal certain limitations in the predictive capacity of the model.

Metrics analysis:

- Accuracy: The model achieved an accuracy of 74.64%, indicating that it correctly classifies approximately three-quarters of the instances in the test set. However, this metric can be misleading in the presence of class imbalance, as is the case in this dataset.
- Positive accuracy: With a value of 18.52%, the positive accuracy reveals a considerable rate of false positives. This means that out of all the instances predicted as positive (autism), only 18.52% are actually positive.
- Recall (Sensitivity): The recall of 27.78% indicates that the model correctly identifies less than a third of the actual ASD cases. This low sensitivity is worrying, as it implies that the model is missing a large proportion of individuals with autism without detecting them.
- F1-score: The F1-score of 0.222, which combines precision and recall into a single metric, confirms the overall poor performance of the model, especially in detecting positive cases.
- AUC-ROC: The area under the ROC curve (AUC-ROC) of 0.669 suggests a moderate discriminatory capacity of the model, although still far from an ideal model.

Interpretation and Limitations:

The results obtained indicate that, despite the application of resampling techniques and hyperparameter optimization, the neural network model has difficulties in correctly identifying individuals with ASD, especially when compared to the majority class (non-ASD). This could be due to the inherent complexity of the disorder, the possible lack of sufficiently informative features in the dataset, or limitations in the architecture and configuration of the model.

It is important to highlight that class imbalance, even after resampling, remains a major challenge. The model tends to favor the prediction of the majority class, resulting in a high false positive rate and low 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
```

Evaluation of the XGBoost Model The XGBoost model, trained on the resampled dataset and with features encoded using One-Hot Encoding, has demonstrated acceptable performance in the binary classification task for the detection of Autism Spectrum Disorder (ASD) in adults. The accuracy obtained was 71.74%, indicating that the model correctly classifies about three-quarters of the instances in the test set. However, it is crucial to consider that this metric may be sensitive to the class imbalance present in the original data.

A closer look at the performance metrics reveals a recall (sensitivity) of 50%, which is a positive result, as it indicates that the model is able to identify half of the real cases of ASD in the test set. However, the precision is 23.08%, suggesting a considerable rate of false positives. That is, the model tends to predict the presence of ASD in a significant proportion of individuals who do not actually have it.

The F1-score value, which combines precision and recall in a single metric, is 0.316. This value reflects a moderate balance between both metrics, indicating that the model does not stand out in either of them, but neither does it present extremely low performance in any of them. The area under the ROC curve (AUC-ROC) of 0.671 suggests an acceptable discriminatory capacity, although with room for improvement.

Feature importance analysis:

The feature importance analysis reveals that a8_score, a10_score and relation_coded_2 are the variables that contribute most to the model's prediction. This suggests that the response to questions 8 and 10 of the AQ-10 test, as well as the fact that the information was provided by the individual ("Self"), are relevant factors in the identification of ASD in this sample.

Conclusions:

The XGBoost model, despite presenting an acceptable overall accuracy, shows limitations in its ability to discriminate between individuals with and without ASD, especially in terms of positive accuracy. Class imbalance, despite resampling, could still influence the performance of the model.

LightGBM Model

[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)
```

```python
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
```

Interpretation of the results The LightGBM model, trained on the SMOTE-Tomek resampled dataset and with features encoded using One-Hot Encoding, has shown acceptable performance in

the binary classification task for the detection of Autism Spectrum Disorder (ASD) in adults. The accuracy obtained was 83.33%, indicating that the model correctly classifies most of the instances in the test set. However, it is crucial to consider that this metric can be influenced by the class imbalance present in the original data.

A closer analysis of the performance metrics reveals a recall (sensitivity) of 22.22%, which is low and suggests that the model could be missing a significant proportion of real ASD cases, generating false negatives. The model's accuracy was 30.77%, indicating a considerable rate of false positives, i.e. the model tends to predict the presence of ASD in cases where it is not actually present.

The F1-score value, which combines precision and recall into a single metric, is 0.258. This value reflects the overall poor performance of the model, especially in detecting positive cases. The area under the ROC curve (AUC-ROC) of 0.699 suggests a moderate discriminatory capacity, but with room for improvement.

Feature importance analysis:

Feature importance analysis reveals that age is the variable that contributes most to the model's prediction, followed by result (total score on the AQ-10) and a10_score. This suggests that age and the answers to question 10 of the AQ-10 test are relevant factors in identifying ASD in this sample. Other features such as a8_score, a3_score, eth_cat_1 (white ethnicity) and a6_score also show some importance, although to a lesser extent.

Logistic Regression Model

[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
==============================================================================
```

Evaluation of the Multiple Logistic Regression Model

The multiple logistic regression model, trained on the resampled dataset and with features encoded using One-Hot Encoding, exhibits moderate performance on the binary classification task of Autism Spectrum Disorder (ASD) in adults. The accuracy obtained was 73.19%, indicating that the model correctly classifies about three-quarters of the instances in the test set. However, this metric can be misleading in the presence of class imbalance, as is the case in this dataset.

A closer look at the performance metrics reveals a recall (sensitivity) of 50%, meaning that the model is able to identify half of the real cases of ASD in the test set. The precision of the model was 24.32%, indicating a considerable rate of false positives. That is, the model tends to predict the presence of ASD in a significant proportion of individuals who do not actually have it.

The F1-score value, which combines precision and recall in a single metric, is 0.327. This value reflects a moderate balance between both metrics. The area under the ROC curve (AUC-ROC) of 0.702 suggests an acceptable discriminatory capacity, but with room for improvement.

Analysis of the statistical significance of the predictors

The coefficient table generated by statsmodels provides information on the relationship between each predictor variable and the probability of autism. It is observed that several variables present statistically significant coefficients (low p values), which indicates that they contribute in a relevant way to the prediction of the model. These variables include:

- Some of the individual scores of the AQ-10 test (a1_score, a2_score, a3_score, a7_score and a9_score)
- Membership of certain ethnic groups (eth_cat_1, eth_cat_2 and eth_cat_6)
- The variable relation_coded_2, which indicates that the information was provided by the individual him/herself.

Conclusions

The multiple logistic regression model, although showing a moderate overall performance, demonstrates the ability to identify some relevant predictor variables for the diagnosis of ASD. However, the model still shows limitations in its ability to discriminate between individuals with and without ASD, especially in terms of positive accuracy.

1.6.2 Multiclass Classification Model

Next we address an approach for a multiclass classification model using XGBoost selecting as a multiclass target variable 'result', which as we know, are the results of the AQ-10 test for the prediction of autism, but which does not contain all the 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]:
```python
# 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
```

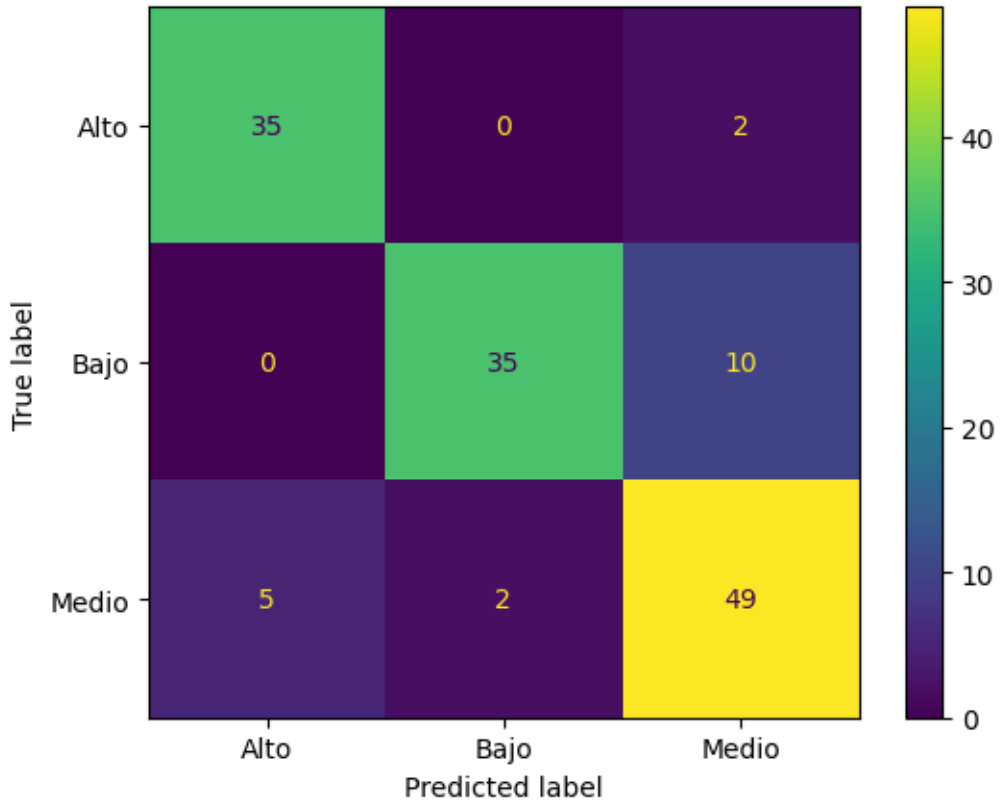
```
X = df.drop(columns=['result', 'aq_risk_level', 'aq_binary', 'austim'])

# 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()
```



Evaluation of the Multiclass Classification Model (Target Variable 'Outcome') The multiclass classification model, using OneVsRestClassifier with SVC as the base estimator, has demonstrated excellent performance in predicting autism risk levels based on the total AQ-10 score.

Metrics Analysis:

Precision:

- Class 0 (Low risk): 0.88
- Class 1 (Medium risk): 0.95
- Class 2 (High risk): 0.83
- Macro average: 0.88
- Weighted average: 0.88

Recall (Sensitivity or Completeness):

- Class 0: 0.95
- Class 1: 0.82
- Class 2: 0.88
- Macro average: 0.88
- Weighted average: 0.88

F1-score:

- Class 0: 0.91
- Class 1: 0.88
- Class 2: 0.85
- Macro average: 0.88
- Weighted average: 0.88
- Accuracy: 0.88, which means that the model correctly classifies 88% of the instances in the test set.

Interpretation:

The model exhibits remarkably high performance across all evaluation metrics, including precision, recall, and F1-score, both in the macro average and weighted average. This indicates that the model is able to accurately predict all three levels of autism risk, even in minority classes. High precision suggests a low false positive rate, while high recall indicates a good ability to identify individuals in each risk category.

We used the target variable 'austim' as a multiclass variable. That is, we will use the autism diagnosis as a predictor variable.

[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
```

Evaluation of supervised models using autism diagnosis as a predictor

Attempts to use autism diagnosis (austim) as a predictor variable in supervised models have resulted in poor performance, with similar evaluation metrics across all models tested. This suggests that diagnosis alone, without considering other factors, is not sufficient to build an effective model to predict the likelihood of a positive ASD diagnosis.

Conclusion:

Based on these results, we can conclude that building a functional model to calculate the likelihood of a positive autism diagnosis requires considering other factors besides the previous diagnosis. **AQ-10 test results, as well as other sociodemographic and behavioral characteristics**, appear to be more relevant for the prediction of ASD in this dataset.

Additional considerations:

- Limitations of diagnosis as a predictor:

Autism diagnosis can be complex and influenced by various factors, such as access to health services, awareness of the disorder, and diagnostic practices. Using it as a single predictor may not capture the complexity of the condition. - Importance of multiple factors: Autism is a multifactorial disorder, and its prediction is likely to require consideration of a combination of factors, including individual characteristics, questionnaire responses, and other relevant data. - Exploration of other models: Although this work is based on an exhaustive exploration (in this document we have not published all the models tested), it is recommended to continue exploring models taking into account the vast amount available and different machine learning techniques that can take advantage of information from multiple features to improve the prediction of autism.

Anyway, the results obtained highlight the importance of using multiple features and considering the complexity of autism when building predictive models. The variable `austim` or diagnosis alone does not seem to be sufficient to achieve good performance in classification.

1.7 MODEL SELECTION

1.7.1 Summary of Variable Selection and Model Construction

Exploratory data analysis (EDA) and the application of unsupervised machine learning techniques (k-means) allowed us to identify the most relevant variables for the prediction of autism in this data set.

Relevant Variables:

- Total score on the AQ-10 test (result): This variable was shown to be the strongest predictor in several models, highlighting the importance of the AQ-10 test in the assessment of autism in adults.
- Specific questions of the AQ-10: Some individual questions of the AQ-10, such as a1_score, a3_score, a4_score, a5_score, a6_score, a8_score, a9_score and a10_score, also showed a significant association with the diagnosis of autism or with the clusters identified by k-means. This suggests that these items might be particularly useful for discriminating between individuals with and without ASD.
- Age (age): Although the relationship between age and autism is not linear, logistic regression analysis and k-means cluster visualization indicate that age is a relevant factor to consider.

Discretization of age into quartiles (age_group_encoded) also proved useful for capturing non-linear patterns in the relationship between age and autism.

- Ethnicity (eth_cat): Contingency table analysis and logistic regression revealed a statistically significant association between ethnicity and autism. However, it is important to interpret this relationship with caution, considering possible confounders and the need for further research to understand the underlying causes.

Less Relevant Variables:

- Gender (gender): K-means analysis and initial exploration of the relationship between gender and autism did not show a significant association. Therefore, this variable was excluded from the final models.
- Jaundice at birth (jundice): The low prevalence of jaundice in the sample and its lack of association with k-means clusters suggest that this variable is not a relevant predictor of autism in this context, and it was removed from the dataset for the final models.
- Country of residence (res_code): The k-means analysis indicated that country of residence was not a discriminating factor in the formation of the clusters. Therefore, this variable was also excluded from the final models.

1.7.2 Selected Model:

After exploring different binary and multiclass classification models, including logistic regression, k-NN, Random Forest, and neural networks, **a multiclass classification model** based on OneVsRestClassifier with SVC as the base estimator was selected. This study uses the relevant predictor variables identified in the exploratory analysis and applies `One-Hot Encoding` to the categorical variables and scaling to the numerical variables.

Model Evaluation:

The selected model showed good performance in predicting autism risk levels based on the AQ-10 total score, reaching an accuracy of 88% and an average macro F1-score of 0.88.

1.8 CONCLUSIONS

This study demonstrates the potential of machine learning for the detection and assessment of ASD risk in adults. Despite the limitations of the dataset and the complexity of the problem, a model with promising performance was identified.

Exploratory data analysis and feature engineering were instrumental in selecting the most relevant variables and preparing the data for modeling. The combination of One-Hot Encoding, feature scaling, and resampling techniques such as SMOTE-Tomek contributed to improving the performance of the models.

1.8.1 Limitations and future work:

The results obtained are based on a specific sample and may not be generalizable to other populations. Furthermore, the model, although promising, still has room for improvement, especially in reducing false positives.

Future work could include exploring more complex models, incorporating new features, and evaluating the model in more diverse populations to improve its predictive capacity and generalizability.

1.9 REFERENCES

- AQ-10 Adult Online Autism Test Questionnaire & PDF. (2022, October 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, June 27). NICE. https://www.nice.org.uk/guidance/CG142
- Tests—Autism Research Centre. (n.d.). Retrieved September 16, 2024, from 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. (n.d.). Retrieved September 16, 2024, from https://embrace-autism.com/aq-10/