Proyecto Integrador Primer Semestre

Representación gráfica georreferenciada de la clasificación de la calidad de planteles educativos con análisis de correlación con planteles vecinos categorizada por variables sociales, económicas y demográficas

Maestría en ciencia de los datos y analítica

16/06/2021

Equipo de desarrollo

- **Camilo Rivera Bedoya**
- **Juan David Corea**
- **Jose Ignacio Escobar**
- **Eliana Marcela Sierra**
- **Daniel Romero Cardona**

¡¡¡Advertencia!!!

- este notebook fue desarrollado en Jupyter notebooks, el abrirlo con otros aplicativos como google colab u otros puede afectar el funconamiento del codigo o los markdowns.

Tabla de contenido

1. ◉ Introduccion

El presente proyecto busca determinar cuál plantel educativo posee mayor calidad partiendo de condiciones de entrada definidas y ubicado en una zona particular y su correlación con planteles cercanos bajo las mismas condiciones, además, de obtener una recomendación de los planteles en un rango de distancia definido que cumplan condiciones de clusterización partiendo de las variables seleccionadas.

Problema a resolver

- ¿Cuál institución educativa presenta mejor calidad partiendo de una ubicación especifica del municipio teniendo en cuenta características sociales, económicas y demográficas?
- ¿Cuál institución educativa presenta mejor calidad en el mismo municipio tomando características sociales y económicas?

Solución propuesta

El analisis se basará en variables que representan criterios reales, como la distancia espacial de un lugar específico respecto a las instituciones, su el rendimiento académico, el estrato socioeconómico entre otras variables propias de cada institución

La solución al problema se realizará de forma gráfica, y busca determinar un cerco de instituciones alrededor de la ubicación ingresada, que para casos prácticos puede ser la vivienda de estudiante donde se pueda filtrar por aquellas instituciones cercanas a las cuales pueda acceder.

Para dar solución a este propósito, nos apoyaremos en modelos de segmentación para poder encontrar clúster de instituciones educativas de acuerdo con características propias, el cálculo de distancias desde la georreferenciación, la creación de métricas ponderadas de criterios a la hora de determinar los cercos, análisis descriptivo-exploratorio para determinar correlaciones o relevancia de variables dentro del análisis.

Metodología para la implementación

El proyecto se realizará en 6 etapas, mediante un sistema de recomendación desarrollado en Python con la base de datos de los resultados de las pruebas saber 11.

Etapa 1: Recolección de la información

Para esta etapa se debe descargar las diferentes bases de datos que serán utilizadas en el análisis. La caracterización de las instituciones, su georreferenciación y variaciones descriptivas adicionales. Al ser datos abiertos y disponibles al pública se hará de forma manual y se cargaran al repositorio de almacenamiento seleccionado.

Etapa 2: Implementación del DataLake

Dado el volumen de información y la necesidad de pre-procesar la información, se decide utilizar el Datalake de AWS (S3). Posterior a la ingesta, se procede a hacer un proceso de ETL vía GLUE que nos permita tener la información alojada en tablas en base datos para facilitar el proceso de consolidación de información.

Etapa 3: Estructuración y preparación de los datos

En dicha etapa se espera hacer un proceso de limpieza, consolidación y validación de la base de datos a utilizar con el fin de tener un único dataset unificado a nivel de institución educativo con todas las variables disponibles para posterior revisión de significancia y pertinencia. Para tal fin, el componente de Athena de AWS se vuelve el más indicado.

Etapa 4: Análisis exploratorio de los datos

En la etapa de exploración se busca hacer un análisis estadístico descriptivo de todas las variables disponibles de instituciones disponibles, desde un enfoque univariado (completitud, tendencia central, significancia, distribución) y multivariado (Correlación total y parcial, variabilidad, explicabilidad) que permita determinar cuáles serán las variables indicadas para proceder a la etapa de modelación e industrialización. Adicionalmente, en esta etapa también se incluye la creación de métricas.

Etapa 5: Modelamiento del sistema de recomendación

En primera instancia se busca desarrollar un modelo de corte no supervisado para identificar clúster o grupos de instituciones educativas que tengan características similares, y de esta manera poder análisis estos grupos internamente. De allí, que a la hora de seleccionar la institución más adecuada se pueda acotar la búsqueda entregando el clúster con más cercanía a las características solicitadas. Para ello se busca testear modelos como:

- K-Means
- DBSCAN
- Mean Shift
- AGNES, entre otros

Posterior al modelamiento se busca hacer una evaluación del modelo óptimo, de ahí que revisando en la literatura nos apoyaremos en validaciones internas y externas, donde en la primera se calculara la cohesión (Minimización de la distancia entre integrantes del clúster) y la separación (Maximización de distancia entre los grupos encontrados).

Entendiendo el comportamiento o las particularidades de las instituciones, se propone una aproximación de sistema de recomendación donde se tenga en cuenta una la geolocalización de referencia, así como algunas variables socioeconómicas que permitan acotar las opciones, de manera que al tener el claro los clústeres que más se ajusten, se tengan también la distancia a las instituciones más cercanas que se encuentran dentro de dichos segmentos.

Etapa 6: Implementación de la interfaz de usuario (FrontEnd)

Basándonos en la API suministrada por las librerías de las aplicaciones Gis para el lenguaje Python se construirá una aplicación web que permitirá ingresar los parámetros de entrada definidos por el modelo y mostrará el mapa con las instituciones que cumplen con la condición de cercanía y clusterización. Además, listará ordenadamente estas instituciones partiendo de la calificación de calidad en resultados en las pruebas saber 11, su grafica de distribución de resultados desde el año 2010 de sus estudiantes.

---

2. ◉ Etapa 1: Recolección de la información

Fuentes de datos

Las fuentes de datos para el desarrollo del proyecto se obtuvieron de las páginas de ICFES y de Datos Abiertos Colombia del Ministerio de Tecnologías de la Información y las Comunicaciones, ambas fuentes de información pública disponibles para proposito general, con las siguientes caracteristicas

El repositorio de datos, DataICFES, contiene información de las pruebas que desarrolla el ICFES. Además, pone a disposición de los investigadores, estudiantes, comunidad educativa y comunidad general, los instrumentos necesarios para adelantar investigaciones que estén orientadas al mejoramiento de la calidad educativa. El repositorio de DATOS ABIERTOS COLOMBIA tiene como objetivo promover y habilitar las condiciones para el uso de los datos y en la actualidad cuenta con mas de 5526 datos disponibles para investigar, desarrollar aplicaciones, crear visualizaciones e historias.

- ICFES (https://www.icfes.gov.co/)

1. **BASE DE DATOS 1** ----------
2. **BASE DE DATOS 2** ----------

- Datos Abiertos (https://www.datos.gov.co/)

- **BASE DE DATOS 3** ----------

---

3. ◉ Etapa 2: Implementación del DataLake

1. Crear bucket
2. Creacion de Zonas (Carpetas)
3. Cargar archivos

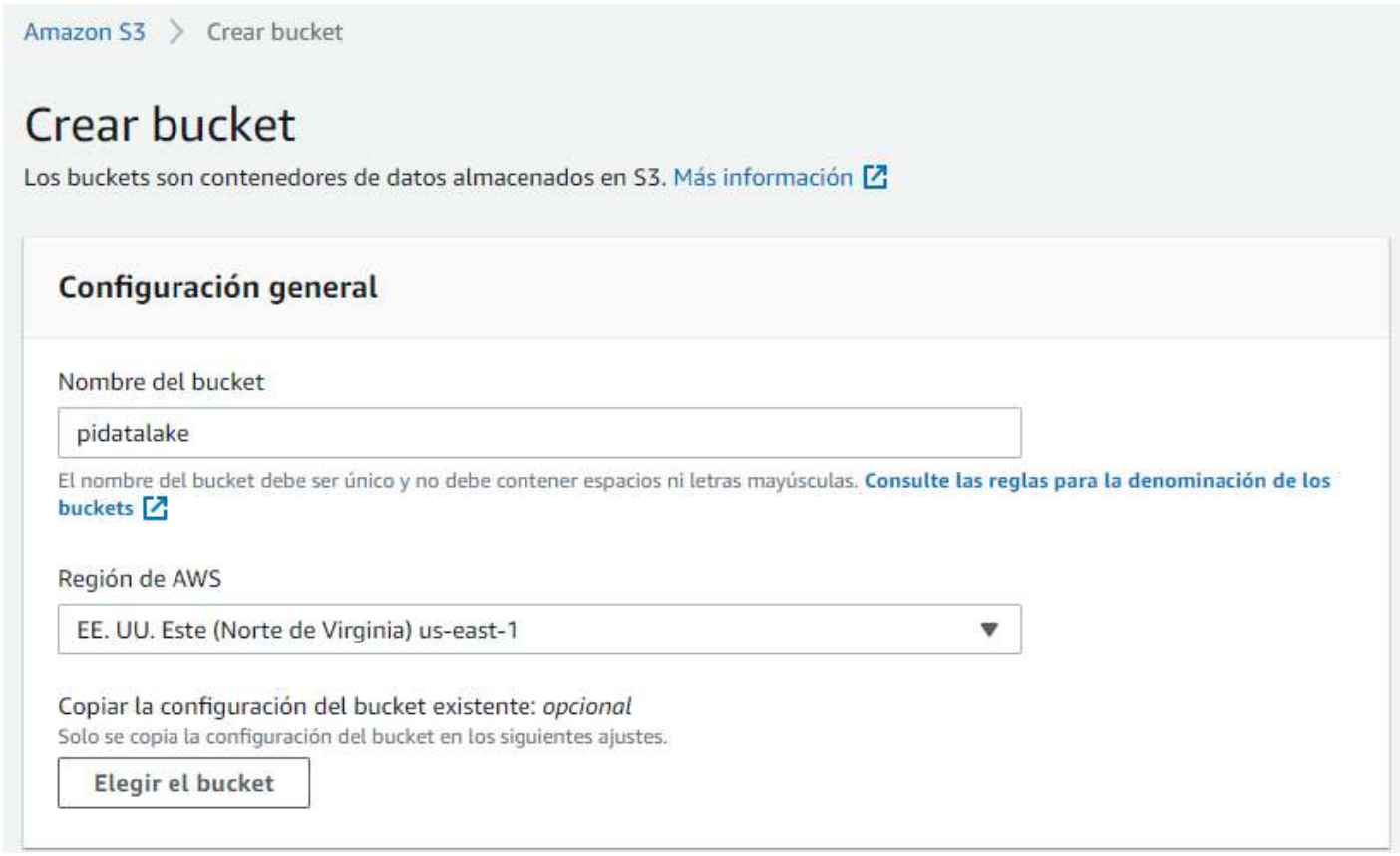
---

se implementara un Data Lake con la siguiente estructura:

- **Landing Zone:** en esta zona se cargan todos los archivos (BD) necesarios para la realizacion del proyecto
- **Raw Zone:** ----------
- **Refined Zone:** ----------
- **Trusted Zone:** ----------

Para Esto nos apoyamos en los recursos de AWS, con la herramienta **S3**.

3.1. ◎ Crear Bucket

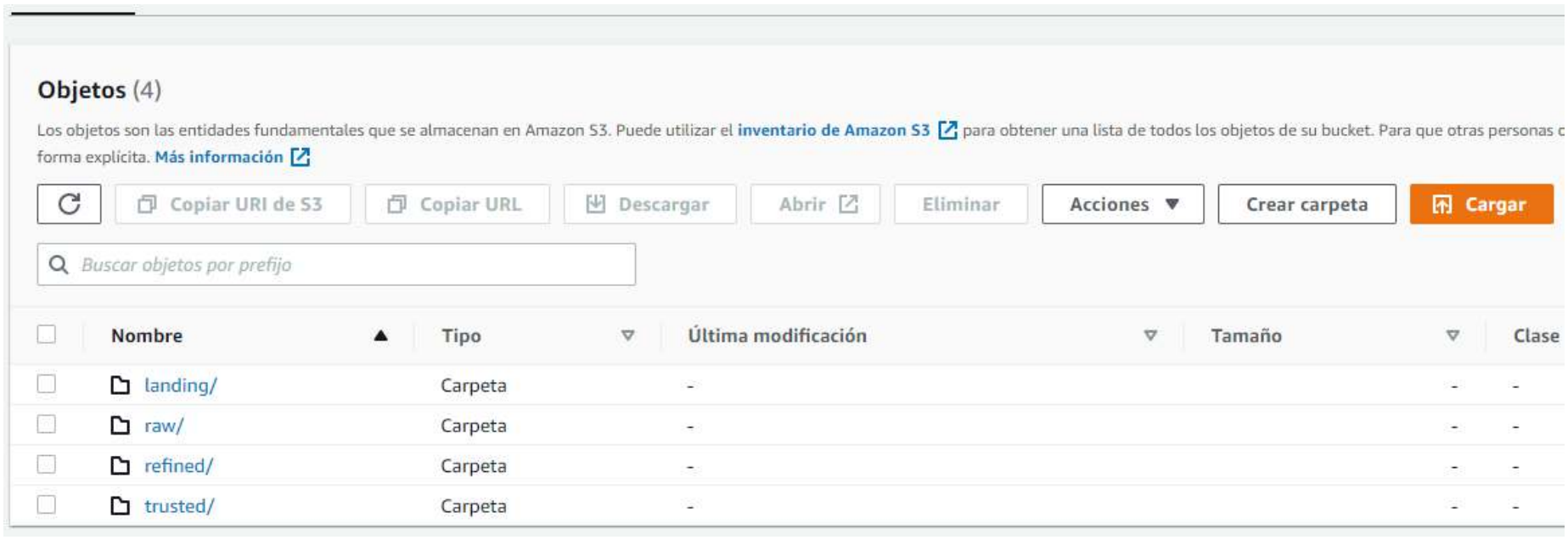


Primero creamos un bucket en S3 con el nombre "**pidatalake**"

Caracteristicas:

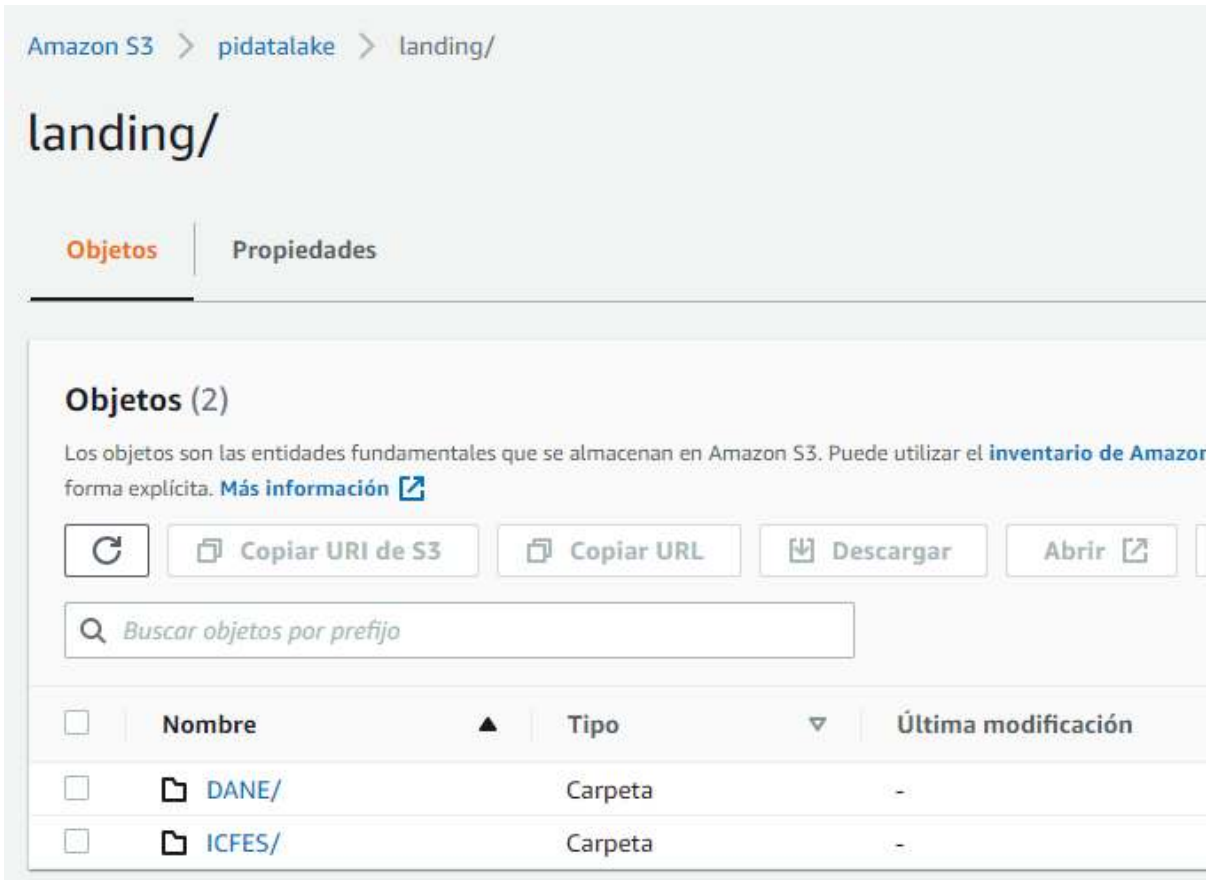
- **Nombre:** pidatalake
- **Región:** EE. UU. Este (Norte de Virginia) us-east-1
- **Configuración de bloqueo de acceso publico:** Ningun bloqueo (publico para pagina web)
- **Control de versiones:** Desactivado
- **Etiquetas:** Ninguna etiqueta
- **Cifrado:** Desactivado
- **Bloqueo de Objetos:** Desactivado

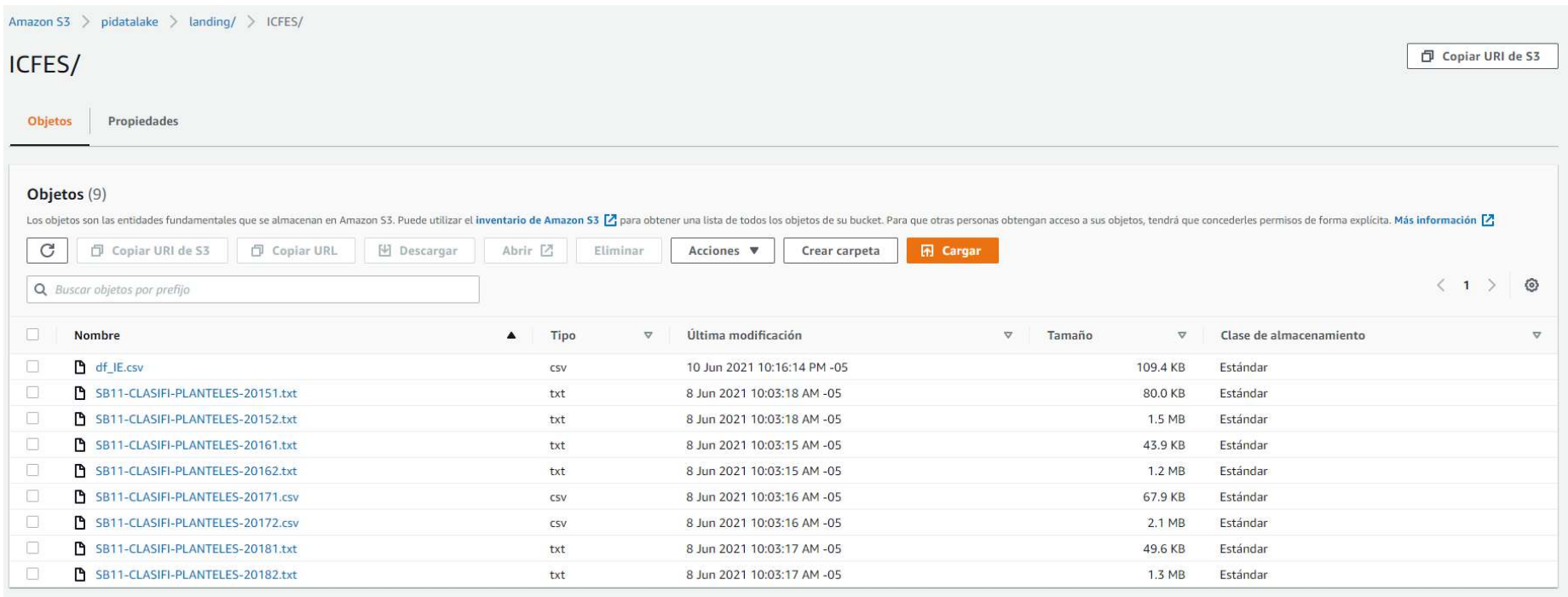
3.2. ◎ Creacion de Zonas (Carpetas)

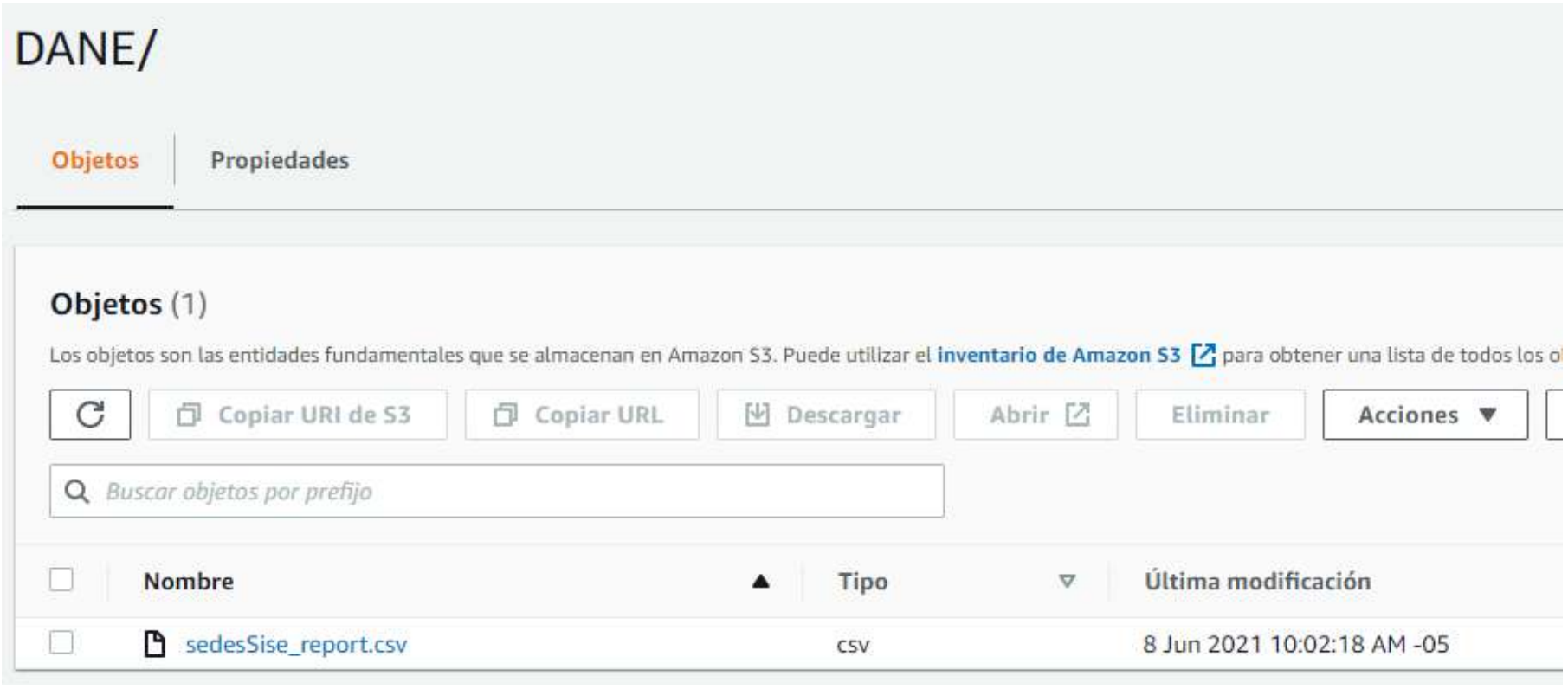


3.3. ◎ Cargar archivos

los archivos se cargan directamente a la landing zone en dos carpetas: **ICFES** y **DANE**



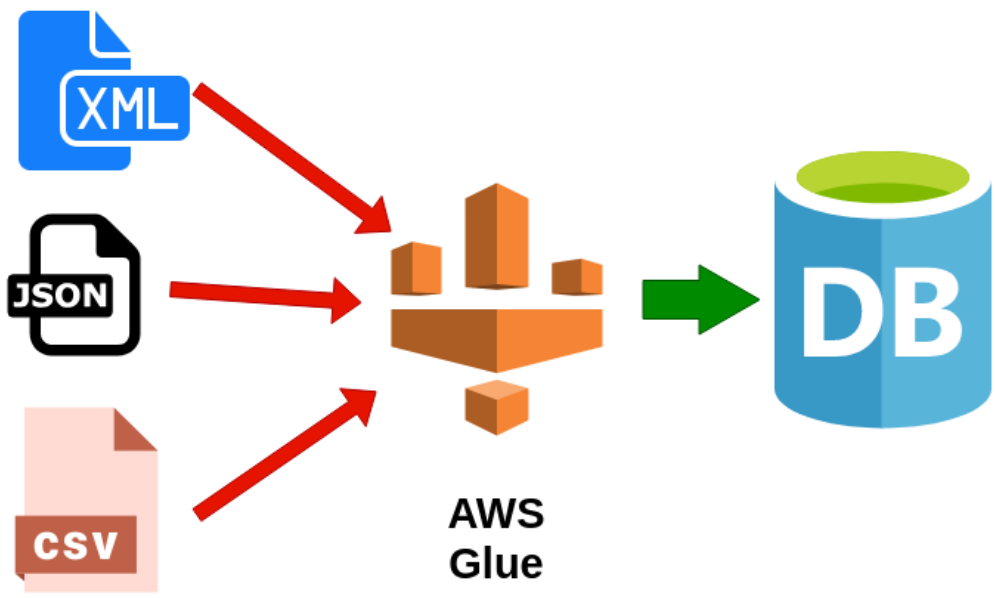




4. Etapa 3: Estructuración y preparación de los datos

Una vez estructurado nuestro Buckets en **Amazon S3** el siguiente paso es convertir estos archivos en información(interpretarlos), para esto el servicio de **AWS Glue** nos permite generar rastreadores, estos rastreadores se deben configurar para que puedan identificar el esquema de los archivos almacenados en **S3** para que estos se puedan interpretar como tablas que se almacenan en una base de datos. Ya que en nuestro datalake contamos con una variedad amplia de fuentes y de formatos es necesario la creación de diferentes bases de datos, tablas y por ende rastreadores

Una ves se realiza este proceso la base de datos esta lista y disponible para ser consultado desde diferentes etapas del desarrollo.



---------- IMAGEN BASE DE DATOS----------

---

5. ◉ Etapa 4: Análisis exploratorio de los datos

j. CARACTER_IE

5.1. ◎ Librerias

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from os import listdir
from scipy.spatial.distance import cdist # distancias entre matrices
from sklearn import cluster #algoritmos de clasificación
from sklearn.metrics import silhouette_score

import sys
import warnings

import ipywidgets as widgets
import seaborn as sns
import statsmodels.api as sm
from IPython.display import HTML, display
from matplotlib.ticker import PercentFormatter
from scipy import stats
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import OneHotEncoder
from statsmodels.graphics.mosaicplot import mosaic
from statsmodels.stats import diagnostic
import scipy.stats as sta

import unicodedata

# Ejecición archivo .py con funciones analisis descriptivos
#import descriptive_PI as explo

import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

import seaborn as sns
```

5.2. ◎ Funciones

In [2]:
```
def wavg(group, avg_name, weight_name):
    ''' Funcion para calcular promedio ponderado de indicadores
    por año e institucion'''
    d = group[avg_name]
    w = group[weight_name]
    try:
        return (d * w).sum() / w.sum()
    except ZeroDivisionError:
        return d.mean()
```

In [3]:
```python
def k_selection(dataframe, min_k, max_k):
    '''
    Función que corre múltiples modelos k-means con diferentes
    números de clusters, calcula y grafica la distancia a los centroides
    de los clusters de cada punto para determinar el número de clusters
    adecuado aplicando diferentes métodos.

    Input:
    ------------------------
    dataframe: pd.DataFrame
        DataFrame con todos los datos para entrenar el modelo.
    min_k: Int
        Número mínimo de clusters a tener en cuenta en las iteraciones.
    max_k: Int
        Número máximo de clusters a tener en cuenta en las iteraciones.

    Output:
    ------------------------
    k_mean_algs: List
        Lista de objetos algoritmo K-means.
    k_mean_res: List
        Lista de resultados del algoritmo K-means.
    '''
    cluster_nums = range(min_k, max_k + 1)# Lista de número de clusters
    k_mean_algs = [cluster.KMeans(n_clusters = k, random_state = 123) for k in cluster_nums]# Lista de objetos algoritmo
    k_mean_res =[]
    sil = []
    for alg in k_mean_algs:
        k_mean_res.append(alg.fit(dataframe))
        labels = alg.labels_
        sil.append(silhouette_score(dataframe, labels, metric = 'euclidean'))
    #k_mean_res = [alg.fit(dataframe) for alg in k_mean_algs]# Lista de resultados del algoritmo K-means
    centroids = [res.cluster_centers_ for res in k_mean_res]# Lista con los centroides por cada valor de k
    # Lista que almacena la distancia euclidea entre los puntos y los centroides para cada cas de k
    distances = [cdist(dataframe, centroid, 'euclidean') for centroid in centroids]
    # Get the closest centroid (and the corresponding distance)
    min_indices = [np.argmin(distance, axis = 1) for distance in distances]
    min_distances = [np.min(distance, axis = 1) for distance in distances]
    avg_sum_squares = [sum(dist ** 2) / dataframe.shape[0] for dist in min_distances]# Calculo de la distancia cuadrada p
    # Gráfica de codo
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.plot(cluster_nums, avg_sum_squares, 'b*-')
    plt.grid(True)
    plt.xlabel('Numero de clusters')
    plt.ylabel('Error cuadrado promedio')
    plt.title('Grafica de Codo')
    plt.show()
    plt.plot(cluster_nums,sil, 'b*-')
    plt.grid(True)
    plt.xlabel('Numero de clusters')
    plt.ylabel('Silhouette Value')
    plt.title('Silhouette Method')
    plt.show()
    return (k_mean_algs, k_mean_res)
```

In [4]:
```python
def tabla_cluster(data,variable):
    '''
    Función para hacer tabla donde se cuentan los puntos por categoría
    según una variable determinada.

    Input:
    ------------------------
    data: pd.DataFrame
        DataFrame con todos los datos.
    variable: String
        Variable por la cual se agrupa en la tabla.

    Output:
    ------------------------
    tbl_pvt: pd.DataFrame
        Tabla agrupada con cuenta de fronteras por categoría.
    '''
    tbl = data[[variable, 'Cluster']]
    tbl_grp = tbl.groupby(tbl.columns.tolist(), as_index = False).size()
    tbl_frm = tbl_grp
    tbl_frm.columns = [variable, 'Cluster', 'Cuenta']
    tbl_pvt = pd.pivot_table(tbl_frm, index = variable, columns = 'Cluster', values = 'Cuenta').fillna(0).astype(int)
    return (tbl_pvt,tbl_frm)
```

In [5]:
```
def cluster_scatter_plot(tbl_frm,variable):
    '''
    Función para graficar la cantidad de fronteras por cluster
    y por una variable.

    Input:
    ------------------------
    tbl_frm: pd.DataFrame
        Tabla con la información para la gráfica.
    variable: String
        Variable contra la cual se graficara, toma dos valores
        'Nivel_Tension' o 'Tipo'.
    '''
    tbl_frm['Cuenta'] = ((tbl_frm.loc[:,'Cuenta']/(tbl_frm.groupby(tbl_frm.Cluster).transform('sum')).loc[:,'Cuenta'])*5
    tbl_frm['Cuenta'] = tbl_frm['Cuenta'].astype('float').round(0).astype('int')
    type_uniq, type_n = np.unique(tbl_frm[variable], return_inverse = True)
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.scatter(x = tbl_frm['Cluster'], y = type_n, s = tbl_frm['Cuenta'])
    ax.set(yticks = np.unique(type_n), yticklabels = type_uniq)
    plt.show()
```

In [6]:
```
def print_one_way_counts_table(col):
    col= pd.Series(np.where(col.isnull(), 'Sin dato', 'Con dato'))
    #freq_table = one_way_table(col=col_count_data, idx_name=idx_name)
    if col is None:
        raise TypeError("Verifique que haya asignado algún objeto al parámetro: 'col'")
    elif col.dtype != 'object':
        raise TypeError("Verifique que el valor asignado al parámetros: 'col', es de tipo: 'object'")
    abs_freq = pd.DataFrame(col.value_counts())
    rel_freq = pd.DataFrame(col.value_counts(normalize=True))
    cum_rel_freq = rel_freq.cumsum()
    freq_table = pd.concat([abs_freq, rel_freq, cum_rel_freq], axis=1)
    freq_table.columns = ['Frec.Abs.','Frec.Rel.', 'Frec.Rel.Acum.']
    #freq_table.index.name = idx_name
    freq_table.sort_values(by=['Frec.Abs.'], ascending=False)
    freq_table=  widgets.HTML(freq_table.style\
                        .format({'Frec.Abs.': '{:,}',
                                 'Frec.Rel.': '{:.2%}',
                                 'Frec.Rel.Acum.': '{:.2%}'})\
                        .set_table_attributes('class="table table-striped"')\
                        .render())
    display(HTML("<h3>Conteo de registros</h3><br>"))
    display(freq_table)
```

In [7]:
```
def print_one_way_table(col):

    if col is None:
        raise TypeError("Verifique que haya asignado algún objeto al parámetro: 'col'")
    elif col.dtype != 'object':
        raise TypeError("Verifique que el valor asignado al parámetros: 'col', es de tipo: 'object'")
    abs_freq = pd.DataFrame(col.value_counts())
    rel_freq = pd.DataFrame(col.value_counts(normalize=True))
    cum_rel_freq = rel_freq.cumsum()
    freq_table = pd.concat([abs_freq, rel_freq, cum_rel_freq], axis=1)
    freq_table.columns = ['Frec.Abs.','Frec.Rel.', 'Frec.Rel.Acum.']
    freq_table.index.name = 'Categorías'
    freq_table.sort_values(by=['Frec.Abs.'], ascending=False)
    display(HTML("<h3>Conteo de frecuencias</h3><br>"))
    display(widgets.HTML(freq_table.style\
                        .format({'Frec.Abs.': '{:,}',
                                 'Frec.Rel.': '{:.2%}',
                                 'Frec.Rel.Acum.': '{:.2%}'})\
                        .set_table_attributes('class="table table-striped"')\
                        .render()))
```

In [8]:
```
def print_central_tendency_measures(col):
    warnings.filterwarnings("ignore")
    notnull_col = col[~np.isnan(col)]
    percentiles = dict(notnull_col.quantile([.05,.25,.5,.75,.95]))
    measures = {}
    measures['Moda'] = notnull_col.mode()[0]
    measures['Media'] = notnull_col.mean()
    try:
        measures['Media Armónica'] = stats.hmean(notnull_col)
    except:
        measures['Media Armónica'] = np.nan
    try:
        measures['Media Geométrica'] = stats.gmean(notnull_col)
    except:
        measures['Media Geométrica'] = np.nan
    measures['Media Cuadrática'] = np.sqrt(np.sum(np.square(notnull_col)) / notnull_col.count())
    measures['Media Trunc.(5%)'] = stats.trim_mean(a=notnull_col, proportiontocut=.05)
    measures['Media IQ'] = stats.trim_mean(a=notnull_col, proportiontocut=.25)
    measures['Media Wins.(5%)'] = np.mean(stats.mstats.winsorize(notnull_col, limits=0.05))
    measures['Trimedia'] = (percentiles[.25] + 2 * percentiles[.5] + percentiles[.75]) / 4
    measures['Mediana'] = np.median(notnull_col)
    measures['Mid Range'] = (notnull_col.min() + notnull_col.max()) / 2
    measures['Mid Hinge'] = (percentiles[.25] + percentiles[.75]) / 2
    statistics = pd.DataFrame.from_dict(data=measures, orient='index', columns=['Resultado'])
    statistics.index.names = ['Medida']
    display(HTML("<h3>Tendencia central</h3><br>"))
    display(widgets.HTML(statistics.style\
                         .format({'Resultado':'{:,.2f}'})\
                         .set_table_attributes('class="table table-striped"')\
                         .render()))
```

In [9]:
```
def location_measures(col):

    notnull_col = col[~np.isnan(col)]
    percentiles = dict(notnull_col.quantile([.01,.05,.1,.25,.5,.75,.9,.95,.99]))
    measures = {}
    measures['Mínimo'] = notnull_col.min()
    measures.update({'Percentil ' + str(int(key * 100)):value for (key, value) in percentiles.items()})
    measures['Máximo'] = notnull_col.max()
    statistics = pd.DataFrame.from_dict(data=measures, orient='index', columns=['Resultado'])
    statistics.index.names = ['Medida']
    return widgets.HTML(statistics.style\
                         .format({'Resultado':'{:,.2f}'})\
                         .set_table_attributes('class="table table-striped"')\
                         .render())
```

In [10]:
```
def print_location_measures(col):

    notnull_col = col[~np.isnan(col)]
    percentiles = dict(notnull_col.quantile([.01,.05,.1,.25,.5,.75,.9,.95,.99]))
    measures = {}
    measures['Mínimo'] = notnull_col.min()
    measures.update({'Percentil ' + str(int(key * 100)):value for (key, value) in percentiles.items()})
    measures['Máximo'] = notnull_col.max()
    statistics = pd.DataFrame.from_dict(data=measures, orient='index', columns=['Resultado'])
    statistics.index.names = ['Medida']
    display(HTML("<h3>Posición</h3><br>"))
    display(widgets.HTML(statistics.style\
                         .format({'Resultado':'{:,.2f}'})\
                         .set_table_attributes('class="table table-striped"')\
                         .render()))
```

In [11]:
```
def print_spread_measures(col):

    notnull_col = col[~np.isnan(col)]
    percentiles = dict(notnull_col.quantile([.01,.05,.1,.25,.5,.75,.9,.95,.99]))
    measures = {}
    measures["Desv. Est."] = notnull_col.std()
    measures['Rango'] = notnull_col.max() - notnull_col.min()
    measures['Rango IQ'] = stats.iqr(notnull_col)
    measures['Dif. Abs. Media'] = notnull_col.mad()
    measures['Dif. Abs. Mediana'] = np.median(np.abs(notnull_col - notnull_col.median()))
    measures['Coef. Var.'] = stats.variation(notnull_col)
    measures['QCD'] = (percentiles[0.75] - percentiles[0.25]) / (percentiles[0.75] + percentiles[0.25])
    statistics = pd.DataFrame.from_dict(data=measures, orient='index', columns=['Resultado'])
    statistics.index.names = ['Medida']
    display(HTML("<h3>Dispersión</h3><br>"))
    display(widgets.HTML(statistics.style\
                         .format({'Resultado':'{:,.2f}'})\
                         .set_table_attributes('class="table table-striped"')\
                         .render()))
```

In [12]:
```
def print_shape_measures(col):

    notnull_col = col[~np.isnan(col)]
    measures = {}
    measures['Asimetría'] = col.skew()
    measures['Exc.Curtosis'] = col.kurtosis()
    statistics = pd.DataFrame.from_dict(data=measures, orient='index', columns=['Resultado'])
    statistics.index.names = ['Medida']
    display(HTML("<h3>Forma</h3><br>"))
    display(widgets.HTML(statistics.style\
                         .format({'Resultado':'{:,.2f}'})\
                         .set_table_attributes('class="table table-striped"')\
                         .render()))
```

In [13]:
```
def univar_histogram_plot(col):

    notnull_col = col[~np.isnan(col)]
    plt.style.use('default')
    fig, ax1 = plt.subplots(figsize=(4.8, 3.4))
    sns.distplot(notnull_col, hist=True, kde=True)
    plt.axvline(notnull_col.mean(), color='green', linestyle='dashed', linewidth=2, label="Media")
    plt.axvline(notnull_col.median(), color='blue', linestyle='dashed', linewidth=2, label="Mediana")
    plt.title('Histograma y densidad', fontweight='bold', fontsize=13, fontfamily='arial')
    ax1.xaxis.set_major_locator(plt.MaxNLocator(5))
    ax1.set_xticklabels(['{:,.2f}'.format(x) for x in ax1.get_xticks()], fontsize=10, fontfamily='arial')
    plt.yticks(fontsize=10, fontfamily='arial')
    plt.xlabel(notnull_col.name, fontweight='bold', fontsize=11, fontfamily='arial')
    plt.ylabel("Densidad", fontweight='bold', fontsize=11, fontfamily='arial')
    leg=plt.legend(loc='upper right', title='Estadísticos')
    plt.setp(leg.get_title(), fontweight='bold', fontsize=10, fontfamily='arial')
    plt.setp(leg.get_texts(), fontsize=9, fontfamily='arial')
    plt.show()
```

In [14]:
```
def univar_violin_plot(col):

    notnull_col = col[~np.isnan(col)]
    plt.style.use('default')
    fig, ax1 = plt.subplots(figsize=(5.0, 3.4))
    sns.violinplot(x=notnull_col, palette='Blues')
    plt.title('Diagrama de Violín', fontweight='bold', fontsize=13, fontfamily='arial')
    ax1.xaxis.set_major_locator(plt.MaxNLocator(5))
    ax1.set_xticklabels(['{:,.2f}'.format(x) for x in ax1.get_xticks()], fontsize=10, fontfamily='arial')
    plt.xlabel(notnull_col.name, fontweight='bold', fontsize=11, fontfamily='arial')
    plt.show()
```

In [15]:
```
def univar_box_plot(col):

    notnull_col = col[~np.isnan(col)]
    plt.style.use('default')
    fig, ax1 = plt.subplots(figsize=(5.0, 3.4))
    sns.boxplot(x=notnull_col, palette='Blues')
    plt.axvline(notnull_col.mean(), color='green', linestyle='dashed', linewidth=2, label='Media')
    plt.title('Diagrama de caja y bigotes', fontweight='bold', fontsize=13, fontfamily='arial')
    ax1.xaxis.set_major_locator(plt.MaxNLocator(5))
    ax1.set_xticklabels(['{:,.2f}'.format(x) for x in ax1.get_xticks()], fontsize=10, fontfamily='arial')
    plt.xlabel(notnull_col.name, fontweight='bold', fontsize=11, fontfamily='arial')
    leg=plt.legend(loc='upper right', title='Estadístico')
    plt.setp(leg.get_title(), fontweight='bold', fontsize=10, fontfamily='arial')
    plt.setp(leg.get_texts(), fontsize=9, fontfamily='arial')
    plt.show()
```

In [16]:
```
def univar_donut_plot(col):
    col_counts = col.value_counts(ascending=False)
    plt.style.use('default')
    plt.figure(figsize=(5.2, 3.6))
    explode=np.repeat(0.03, len(col_counts))
    centre_circle = plt.Circle((0,0), 0.70, fc='white')
    plt.pie(x=col_counts, explode=explode, labels=None, autopct='%1.1f%%', textprops={'fontsize': 10, 'fontfamily': 'ari
    plt.gcf().gca().add_artist(centre_circle)
    plt.title('Diagrama circular', fontweight='bold', fontsize=13, fontfamily='arial')
    leg=plt.legend(labels=col_counts.index, loc="lower left", title=col.name, framealpha=0.4)
    plt.setp(leg.get_title(), fontweight='bold', fontsize=10, fontfamily='arial')
    plt.setp(leg.get_texts(), fontsize=9, fontfamily='arial')
    plt.axis('equal')
    plt.show()
```

In [17]:
```
def univar_pareto_plot(col):
    col_counts = col.value_counts(ascending=False)
    x = col_counts.index
    y = col_counts.values
    xlabel=col.name
    ylabel='Frecuencia'
    weights = col_counts / col_counts.sum()
    cumsum = weights.cumsum()
    plt.style.use('default')
    fig, ax1 = plt.subplots(figsize=(5.0, 3.4))
    sns.barplot(x=x, y=y, hue=x, palette="bright")
    plt.title('Diagrama de pareto', fontweight='bold', fontsize=13, fontfamily='arial')
    ax1.set_xlabel(xlabel, fontweight='bold', fontsize=11, fontfamily='arial')
    ax1.set_ylabel(ylabel, fontweight='bold', fontsize=11, fontfamily='arial')
    ax1.set_yticklabels(['{:,.0f}'.format(x) for x in ax1.get_yticks()], fontsize=10, fontfamily='arial')
    ax1.legend(loc='right', framealpha=0.4)
    ax2 = ax1.twinx()
    ax2.plot(x, cumsum, '-ro', alpha=0.8, color='gray')
    ax2.set_ylabel('Frec. Relat. Acum.', fontweight='bold', fontsize=11, fontfamily='arial')
    ax2.set_yticklabels(['{:,.0%}'.format(x) for x in ax2.get_yticks()], fontsize=10, fontfamily='arial')
    formatted_weights = ['{:,.0%}'.format(x) for x in cumsum]
    for i, txt in enumerate(formatted_weights):
        ax2.annotate(txt, (x[i], cumsum[i]), fontsize=9, fontweight='bold', fontfamily='arial')
    plt.setp(ax1.legend_.get_texts(), fontsize=9, fontfamily='arial')
    plt.xticks([])
    plt.tight_layout()
    plt.show()
```

In [18]:
```
def univar_normal_density(col):

    notnull_col = col[~np.isnan(col)]
    x = np.linspace(notnull_col.min(), notnull_col.max(), 1000)
    plt.style.use('default')
    fig, ax1 = plt.subplots(figsize=(4.8, 3.4))
    sns.distplot(notnull_col, hist=False, kde=True, label='Empírica')
    ax1.plot(x, stats.norm.pdf(x=x, loc=notnull_col.mean(), scale=notnull_col.std()), label='Normal')
    plt.title('Funciones de densidad', fontweight='bold', fontsize=13, fontfamily='arial')
    ax1.xaxis.set_major_locator(plt.MaxNLocator(5))
    ax1.set_xticklabels(['{:,.2f}'.format(x) for x in ax1.get_xticks()], fontsize=10, fontfamily='arial')
    plt.yticks(fontsize=10, fontfamily='arial')
    plt.xlabel(notnull_col.name, fontweight='bold', fontsize=11, fontfamily='arial')
    plt.ylabel("Densidad", fontweight='bold', fontsize=11, fontfamily='arial')
    leg=plt.legend(loc='upper right', title='Distribución')
    plt.setp(leg.get_title(), fontweight='bold', fontsize=10, fontfamily='arial')
    plt.setp(leg.get_texts(), fontsize=9, fontfamily='arial')
    plt.show()
```

In [19]:
```
def univar_qq_norm(col):

    notnull_col = col[~np.isnan(col)]
    fig = sm.qqplot(data=notnull_col, fit=True, marker='.', markerfacecolor='w', markeredgecolor='tab:blue', markersize=
    sm.qqline(fig.axes[0], line='45', fmt='k-')
    plt.title('Diagrama cuantil-cuantil', fontweight='bold', fontsize=13, fontfamily='arial')
    plt.yticks(fontsize=10, fontfamily='arial')
    plt.xticks(fontsize=10, fontfamily='arial')
    plt.xlabel('Cuantiles teóricos normales', fontweight='bold', fontsize=11, fontfamily='arial')
    plt.ylabel('Cuantiles muestrales', fontweight='bold', fontsize=11, fontfamily='arial')
    fig.set_size_inches(4.8, 3.4, forward=True)
    plt.show()
```

In [20]:
```
def print_normality_tests(col):

    warnings.filterwarnings("ignore")
    notnull_col = col[~np.isnan(col)]
    try:
        sw_stat, sw_pvalue = stats.shapiro(x=notnull_col)
    except:
        sw_stat, sw_pvalue = (np.nan, np.nan)
    try:
        ad_stat, ad_pvalue = diagnostic.normal_ad(x=notnull_col)
    except:
        ad_stat, ad_pvalue = (np.nan, np.nan)
    try:
        ll_stat, ll_pvalue = diagnostic.lilliefors(x=notnull_col, dist='norm')
    except:
        ll_stat, ll_pvalue = (np.nan, np.nan)
    try:
        dp_stat, dp_pvalue = stats.normaltest(a=notnull_col)
    except:
        dp_stat, dp_pvalue = (np.nan, np.nan)
    try:
        jb_stat, jb_pvalue = stats.jarque_bera(x=notnull_col)
    except:
        jb_stat, jb_pvalue = (np.nan, np.nan)
    try:
        cp_stat, cp_pvalue = stats.combine_pvalues(pvalues=[sw_pvalue, ad_pvalue, ll_pvalue, dp_pvalue, jb_pvalue], metho
    except:
        cp_stat, cp_pvalue = (np.nan, np.nan)

    normal_tests = {}
    normal_tests = {'Valores P.': [sw_pvalue, ad_pvalue, ll_pvalue, dp_pvalue, jb_pvalue, cp_pvalue]}
    one_dim_normal_tests = pd.DataFrame(data=normal_tests,
                                        dtype=np.float,
                                        index=['Shapiro-Wilk (SW)','Anderson-Darling (AD)','Lilliefors (LL)','D'Agostino
    one_dim_normal_tests.index.names = ['Prueba (Test)']
    display(HTML("<h3>Pruebas de normalidad</h3><br>"))
    display(widgets.HTML(one_dim_normal_tests.style\
                         .format({'Valores P.':'{:,.2%}'})\
                         .set_table_attributes('class="table table-striped"')\
                         .render()))
```

In [21]:
```
def inter_uncond_descrp_num_var(col):

    tabTab = widgets.Tab()
    outTab = [widgets.Output(), widgets.Output(), widgets.Output(), widgets.Output()]
    tabTab.children = outTab
    tabTab.set_title(0, "Frecuencia")
    tabTab.set_title(1, "TC y Posición")
    tabTab.set_title(2, "Dispersión y Forma")
    tabTab.set_title(3, "Normalidad")

    acTab = widgets.Accordion(children=[tabTab])
    acTab.set_title(0, 'Estadísticos')

    with outTab[0]:
        print_one_way_counts_table(col)
    auxOut1 = [widgets.Output(), widgets.Output()]
    with auxOut1[0]:
        print_central_tendency_measures(col)
    with auxOut1[1]:
        print_location_measures(col)
    with outTab[1]:
        display(widgets.HBox(auxOut1))

    auxOut2 = [widgets.Output(), widgets.Output()]
    with auxOut2[0]:
        print_spread_measures(col)
    with auxOut2[1]:
        print_shape_measures(col)
    with outTab[2]:
        display(widgets.HBox(auxOut2))

    with outTab[3]:
        print_normality_tests(col)

    tabGraf = widgets.Tab()
    outGraf = [widgets.Output(), widgets.Output(), widgets.Output(), widgets.Output(), widgets.Output()]
    tabGraf.children = outGraf
    tabGraf.set_title(0, "Histograma")
    tabGraf.set_title(1, "Caja y bigotes")
    tabGraf.set_title(2, "Violín")
    tabGraf.set_title(3, "Densidad Norm.")
    tabGraf.set_title(4, "QQ Norm.")

    acGraf = widgets.Accordion(children=[tabGraf])
    acGraf.set_title(0, 'Gráficos descriptivos')

    with outGraf[0]:
        univar_histogram_plot(col)
    with outGraf[1]:
        univar_box_plot(col)
    with outGraf[2]:
        univar_violin_plot(col)
    with outGraf[3]:
        univar_normal_density(col)
    with outGraf[4]:
        univar_qq_norm(col)

    out = widgets.Output()
    with out:
        display(acTab)
        display(acGraf)
    display(out)
```

In [22]:
```
def inter_uncond_descrp_cat_var(col):

    tabTab = widgets.Tab()
    outStat = [widgets.Output()]
    tabTab.children = outStat
    tabTab.set_title(0, "Frecuencia")

    actTab = widgets.Accordion(children=[tabTab])
    actTab.set_title(0, 'Estadísticos')

    auxOut1 = [widgets.Output(), widgets.Output()]
    with auxOut1[0]:
        print_one_way_counts_table(col)
    with auxOut1[1]:
        print_one_way_table(col)
    with outStat[0]:
        display(widgets.HBox(auxOut1))

    tabGraf = widgets.Tab()
    outGraf = [widgets.Output(), widgets.Output()]
    tabGraf.children = outGraf
    tabGraf.set_title(0, "Circular")
    tabGraf.set_title(1, "Pareto")

    actGraf = widgets.Accordion(children=[tabGraf])
    actGraf.set_title(0, 'Gráficos descriptivos')

    with outGraf[0]:
        univar_donut_plot(col)
    with outGraf[1]:
        univar_pareto_plot(col)

    out = widgets.Output()
    with out:
        display(actTab)
        display(actGraf)
    display(out)
```

In [23]:
```
def corr_pearson(df):
    f = plt.figure(figsize=(9, 11))
    plt.matshow(df.corr(), fignum=f.number)
    plt.xticks(range(df.select_dtypes(['number']).shape[1]), Num_df.columns, fontsize=8, rotation=45)
    plt.yticks(range(df.select_dtypes(['number']).shape[1]), Num_df.columns, fontsize=8)
    cb = plt.colorbar()
    cb.ax.tick_params(labelsize=14)
    plt.show()
```

In [24]:
```
def Corr_Kendall(df):
    f = plt.figure(figsize=(9, 11))
    plt.matshow(df.corr(method='kendall'), fignum=f.number)
    plt.xticks(range(df.select_dtypes(['number']).shape[1]), Num_df.columns, fontsize=8, rotation=45)
    plt.yticks(range(df.select_dtypes(['number']).shape[1]), Num_df.columns, fontsize=8)
    cb = plt.colorbar()
    cb.ax.tick_params(labelsize=14)
    plt.show()
```

In [25]:
```
def Corr_Spearman(df):
    corr_sper = pd.DataFrame(sta.spearmanr(Num_df).correlation, columns= Num_df.columns, index=Num_df.columns)
    f = plt.figure(figsize=(9, 11))
    plt.matshow(corr_sper, fignum=f.number)
    plt.xticks(range(df.select_dtypes(['number']).shape[1]), Num_df.columns, fontsize=8, rotation=45)
    plt.yticks(range(df.select_dtypes(['number']).shape[1]), Num_df.columns, fontsize=8)
    cb = plt.colorbar()
    cb.ax.tick_params(labelsize=14)
    plt.show()
```

In [26]:
```
def print_corr_var(df):
    tabTab = widgets.Tab()
    outTab = [widgets.Output(), widgets.Output(), widgets.Output()]
    tabTab.children = outTab
    tabTab.set_title(0, "Pearson")
    tabTab.set_title(1, "Spearman")
    tabTab.set_title(2, "Kendall")


    acTab = widgets.Accordion(children=[tabTab])
    acTab.set_title(0, 'Matriz correlación')

    with outTab[0]:
        corr_pearson(df)
    with outTab[1]:
        Corr_Spearman(df)
    with outTab[2]:
        Corr_Kendall(df)

    out = widgets.Output()
    with out:
        display(acTab)
    display(out)
```

In [27]:
```
def outliers(df):
    tabTab = widgets.Tab()
    outTab = [widgets.Output(), widgets.Output(),widgets.Output()]
    tabTab.children = outTab
    tabTab.set_title(0, "N. Manhattan")
    tabTab.set_title(1, "N.Euclidea")
    tabTab.set_title(2, "N. Frobenius")

    acTab = widgets.Accordion(children=[tabTab])
    acTab.set_title(0, 'Identificacion I.E Atipicas')

    auxOut0 = [widgets.Output(), widgets.Output()]
    with auxOut0[0]:
        display(HTML("<h3>TOP Mejores puntajes </h3><br>"))
        display(df.loc[Dist_1Out.index][['COLE_INST_NOMBRE','INDICE_TOTAL','EVALUADOS_ULTIMOS_3','INDICE_AJUST','COLE_CA
    with auxOut0[0]:
        display(HTML("<h3>TOP peores puntajes </h3><br>"))
        display(df.loc[Dist_1Out.index][['COLE_INST_NOMBRE','INDICE_TOTAL','EVALUADOS_ULTIMOS_3','INDICE_AJUST','COLE_CA
    with outTab[0]:
        display(widgets.HBox(auxOut0))


    auxOut1 = [widgets.Output(), widgets.Output()]
    with auxOut1[0]:
        display(HTML("<h3>TOP Mejores puntajes </h3><br>"))
        display(df.loc[Dist_2Out.index][['COLE_INST_NOMBRE','INDICE_TOTAL','EVALUADOS_ULTIMOS_3','INDICE_AJUST','COLE_CA
    with auxOut1[0]:
        display(HTML("<h3>TOP peores puntajes </h3><br>"))
        display(df.loc[Dist_2Out.index][['COLE_INST_NOMBRE','INDICE_TOTAL','EVALUADOS_ULTIMOS_3','INDICE_AJUST','COLE_CA
    with outTab[1]:
        display(widgets.HBox(auxOut1))

    auxOut2 = [widgets.Output(), widgets.Output()]
    with auxOut2[0]:
        display(HTML("<h3>TOP Mejores puntajes </h3><br>"))
        display(df.loc[Dist_FroOut.index][['COLE_INST_NOMBRE','INDICE_TOTAL','EVALUADOS_ULTIMOS_3','INDICE_AJUST','COLE_
    with auxOut2[0]:
        display(HTML("<h3>TOP peores puntajes </h3><br>"))
        display(df.loc[Dist_FroOut.index][['COLE_INST_NOMBRE','INDICE_TOTAL','EVALUADOS_ULTIMOS_3','INDICE_AJUST','COLE_
    with outTab[2]:
        display(widgets.HBox(auxOut1))

    out = widgets.Output()
    with out:
        display(acTab)
    display(out)
```

5.3. ◎ Lectura Base de Datos - AWS

5.4. ◎ Lectura Base de Datos - Local

1. Sedes
2. Clasificacion Planteles
3. Puntaje estudiantes por Plantel
4. Unificar Base de Datos
5. Limpieza Base de datos

en caso de que se ejecute este codigo en local se deja este fragmento de codigo el cual carga los archivos y realiza la transformacion para obtener una unica base de datos identica a la obtenida desde AWS para poder trabajar con el notebook.

5.4.1 ◐ Sedes

In [28]:
```
sedes = pd.read_csv('..\Datos\sedesSise_report.csv', encoding='latin-1', index_col= False,decimal=',')
sedes = sedes[['COD_COL','NOMBRE_DEPARTAMENTO','NOMBRE_MUNICIPIO','ZONA','LATITUD','LONGITUD']]
sedes = sedes[sedes['NOMBRE_DEPARTAMENTO'] == 'ANTIOQUIA']
sedes = sedes[sedes['LATITUD'] != 0]
sedes = sedes[sedes['LATITUD'] != np.nan]
sedes = sedes[sedes['LONGITUD'] != 0]
sedes = sedes[sedes['LONGITUD'] != np.nan]
sedes.reset_index(drop = True)
sedes.head(3)
```

Out[28]:

	COD_COL	NOMBRE_DEPARTAMENTO	NOMBRE_MUNICIPIO	ZONA	LATITUD	LONGITUD
42	4.050010e+11	ANTIOQUIA	MEDELLIN	Rural	6.181654	-75.642444
43	4.050010e+11	ANTIOQUIA	MEDELLIN	Rural	6.252417	-75.557074
45	4.050010e+11	ANTIOQUIA	MEDELLIN	Rural	6.175561	-75.644494

Variables	Descripción	Categoría
COD_COL	Codigo unico del colegio	Categórica
NOMBRE_DEPARTAMENTO	Departamento del colegio	Categorica
NOMBRE_MUNICIPIO	Municipio del colegio	Categórica
ZONA	Zona Colegio (Rural, Urbano)	Categórica
LATITUD	Latitud ubicacion colegio	Numerica
LONGITUD	Longitud ubicacion colegio	Numerica

5.4.2 ◐ Clasificacion Planteles

In [29]:
```
clasificacion = pd.read_csv('..\Datos\SB11-CLASIFI-PLANTELES-20182.txt', delimiter = '\t', encoding='latin-1',decimal=',
clasificacion = clasificacion[['COLE_COD_DANE','COLE_INST_NOMBRE','COLE_DEPTO_COLEGIO','COLE_CALENDARIO_COLEGIO',
                               'COLE_GENEROPOBLACION','EVALUADOS_ULTIMOS_3','COLE_NATURALEZA','INDICE_MATEMATICAS',
                               'INDICE_C_NATURALES','INDICE_SOCIALES_CIUDADANAS','INDICE_LECTURA_CRITICA','INDICE_INGLES
                               'INDICE_TOTAL','COLE_CATEGORIA']]
clasificacion = clasificacion.rename(columns={'COLE_COD_DANE': 'COD_COL'})
clasificacion = clasificacion[clasificacion['COLE_DEPTO_COLEGIO']=='ANTIOQUIA']
clasificacion.head(3)
```

Out[29]:

	COD_COL	COLE_INST_NOMBRE	COLE_DEPTO_COLEGIO	COLE_CALENDARIO_COLEGIO	COLE_GENEROPOBLACION	EVALUADOS_ULTIMOS_
0	205790000235	I. E. LA INMACULADA	ANTIOQUIA	A	MI	6
3	105147000568	I. E. JOSE MARIA MUÑOZ FLOREZ	ANTIOQUIA	A	MI	26
4	105147000401	I. E. COLOMBIA	ANTIOQUIA	A	MI	23

Variables	Descripción	Categoría
COD_COL	Codigo unico del colegio	Categórica
COLE_INST_NOMBRE	Nombre del colegio	Categórica
COLE_DEPTO_COLEGIO	Departamento del colegio	Categorica

Variables	Descripción	Categoría
COLE_CALENDARIO_COLEGIO	Tipo de calendario academico	Categórica
COLE_GENEROPOBLACION	Tipo genero de los estudiantes (Mixto,Femenino,Masculino)	Categórica
EVALUADOS_ULTIMOS_3	Numero de estudiantes que presentaron la prueba	Numerica
COLE_NATURALEZA	Naturaleza Colegio (Oficial, No Oficial)	Categorica
INDICE_MATEMATICAS	Resultado prueba en Matematicas	Numerica
INDICE_C_NATURALES	Resultado prueba en Ciencias Naturales	Numerica
INDICE_SOCIALES_CIUDADANAS	Resultado prueba en Sociales	Numerica
INDICE_LECTURA_CRITICA	Resultado prueba en Lectura Critica	Numerica
INDICE_INGLES	Resultado prueba en Ingles	Numerica
INDICE_TOTAL	Resultado prueba total	Numerica
COLE_CATEGORIA	Categioria asignada Colegio (D...A+)	Numerica

5.4.3 ◐ Puntaje estudiantes por Plantel

In [30]:
```
df_puntaje = pd.read_csv('..\Datos\puntaje.csv', delimiter = ';', encoding='latin-1',decimal=',',low_memory=False)
df_puntaje = df_puntaje[['FAMI_ESTRATOVIVIENDA','FAMI_EDUCACIONPADRE','FAMI_EDUCACIONMADRE','ESTU_HORASSEMANATRABAJA','C
df_puntaje = df_puntaje.rename(columns={'COLE_COD_DANE_ESTABLECIMIENTO': 'COD_COL'})
df_puntaje = df_puntaje[df_puntaje['COLE_DEPTO_UBICACION'] == 'ANTIOQUIA']
df_puntaje.head(10)
```

Out[30]:

	FAMI_ESTRATOVIVIENDA	FAMI_EDUCACIONPADRE	FAMI_EDUCACIONMADRE	ESTU_HORASSEMANATRABAJA	COD_COL	COLE_BILINGUE
1	Estrato 1	Primaria incompleta	Primaria incompleta	Entre 11 y 20 horas	205051001339	NaN
11	Estrato 2	Primaria incompleta	Secundaria (Bachillerato) incompleta	Menos de 10 horas	105658000124	N
14	Estrato 2	Secundaria (Bachillerato) incompleta	Secundaria (Bachillerato) completa	0	105088002705	N
42	NaN	NaN	NaN	Menos de 10 horas	105234000531	N
54	Estrato 1	Primaria incompleta	Primaria incompleta	Más de 30 horas	105756000493	N
61	Estrato 3	Técnica o tecnológica completa	Educación profesional completa	Menos de 10 horas	105266000061	N
68	NaN	NaN	NaN	NaN	305001022607	N
79	Estrato 2	Educación profesional completa	Educación profesional completa	0	105490000026	N
93	Estrato 2	Secundaria (Bachillerato) completa	Educación profesional completa	0	305360000040	N
101	Estrato 1	Primaria incompleta	Secundaria (Bachillerato) incompleta	0	105250000169	N

In [31]:
```
df_IE= pd.DataFrame(columns=['MODE_ESTRATOVIVIENDA','IND_EST_TRAB','MODE_EDU_MADRE','MODE_EDU_PADRE','IND_BILINGUE','CARA
df_IE['MODE_ESTRATOVIVIENDA']= df_puntaje.groupby(['COD_COL']).agg({'FAMI_ESTRATOVIVIENDA':pd.Series.mode})['FAMI_ESTRAT
df_IE['IND_EST_TRAB']=df_puntaje.groupby(['COD_COL']).agg({'ESTU_HORASSEMANATRABAJA':pd.Series.mode})['ESTU_HORASSEMANATI
df_IE['MODE_EDU_MADRE']= df_puntaje.groupby(['COD_COL']).agg({'FAMI_EDUCACIONMADRE':pd.Series.mode})['FAMI_EDUCACIONMADRI
df_IE['MODE_EDU_PADRE']= df_puntaje.groupby(['COD_COL']).agg({'FAMI_EDUCACIONPADRE':pd.Series.mode})['FAMI_EDUCACIONPADRI
df_IE['IND_BILINGUE']= df_puntaje.groupby(['COD_COL']).agg({'COLE_BILINGUE':pd.Series.mode})['COLE_BILINGUE'].apply(lambd
df_IE['CARACTER_IE']= df_puntaje.groupby(['COD_COL']).agg({'COLE_CARACTER':pd.Series.mode})['COLE_CARACTER'].apply(lambda
df_IE.head(10)
```

Out[31]:

COD_COL	MODE_ESTRATOVIVIENDA	IND_EST_TRAB	MODE_EDU_MADRE	MODE_EDU_PADRE	IND_BILINGUE	CARACTER_IE
105001000001	Estrato 2	0	Secundaria (Bachillerato) completa	Primaria incompleta	N	ACADÉMICO
105001000043	Estrato 2	0	Secundaria (Bachillerato) completa	Primaria incompleta	N	ACADÉMICO
105001000108	Estrato 2	0	Secundaria (Bachillerato) completa	Secundaria (Bachillerato) completa	N	TÉCNICO/ACADÉMICO
105001000132	Estrato 3	0	Secundaria (Bachillerato) completa	Secundaria (Bachillerato) completa	None	ACADÉMICO
105001000141	Estrato 3	0	Secundaria (Bachillerato) completa	Secundaria (Bachillerato) completa	N	ACADÉMICO
105001000175	Estrato 2	0	Secundaria (Bachillerato) completa	Secundaria (Bachillerato) completa	N	TÉCNICO/ACADÉMICO
105001000205	Estrato 2	0	Secundaria (Bachillerato) completa	Secundaria (Bachillerato) completa	None	ACADÉMICO
105001000256	Estrato 2	0	Secundaria (Bachillerato) completa	Primaria incompleta	N	TÉCNICO/ACADÉMICO
105001000353	Estrato 3	0	Secundaria (Bachillerato) completa	Secundaria (Bachillerato) completa	N	ACADÉMICO
105001000396	Estrato 2	0	Secundaria (Bachillerato) completa	Secundaria (Bachillerato) completa	N	TÉCNICO/ACADÉMICO

Variables	Descripción	Categoría
COD_COL	Codigo unico del colegio	Categórica
MODE_ESTRATOVIVIENDA	Estrato mas comun de los estudiantes del colegio	Categorica
IND_EST_TRAB	Estudiante Trabaja (Si 1, No 0)	Categorica
MODE_EDU_MADRE	Educacion Madre mas comun de los estudiantes del colegio	Categorica
MODE_EDU_PADRE	Educacion Padre mas comun de los estudiantes del colegio	Categorica
IND_BILINGUE	Colegio Bilingue (N, S)	Categorica
CARACTER_IE	Caracter Institucion Educativa (Tecnico, Academico)	Categorica

5.4.4 ◐ Unificar Base de Datos

In [32]:
```
df = pd.merge(clasificacion,sedes,how='inner',on='COD_COL')
df = pd.merge(df,df_IE,how='inner',on='COD_COL')
df.head(3)
```

Out[32]:

	COD_COL	COLE_INST_NOMBRE	COLE_DEPTO_COLEGIO	COLE_CALENDARIO_COLEGIO	COLE_GENEROPOBLACION	EVALUADOS_ULTIMOS_
0	205790000235	I. E. LA INMACULADA	ANTIOQUIA	A	MI	6
1	105147000568	I. E. JOSE MARIA MUÑOZ FLOREZ	ANTIOQUIA	A	MI	26
2	105147000401	I. E. COLOMBIA	ANTIOQUIA	A	MI	23

3 rows × 25 columns

In [33]:
```
numericas = ['INDICE_MATEMATICAS','INDICE_C_NATURALES','INDICE_SOCIALES_CIUDADANAS','INDICE_LECTURA_CRITICA',
             'INDICE_INGLES','INDICE_TOTAL','EVALUADOS_ULTIMOS_3']
df[numericas] = df[numericas].apply(pd.to_numeric)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 968 entries, 0 to 967
Data columns (total 25 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   COD_COL                      968 non-null    int64
 1   COLE_INST_NOMBRE             968 non-null    object
 2   COLE_DEPTO_COLEGIO           968 non-null    object
 3   COLE_CALENDARIO_COLEGIO      968 non-null    object
 4   COLE_GENEROPOBLACION         968 non-null    object
 5   EVALUADOS_ULTIMOS_3          968 non-null    int64
 6   COLE_NATURALEZA              968 non-null    object
 7   INDICE_MATEMATICAS           968 non-null    float64
 8   INDICE_C_NATURALES           968 non-null    float64
 9   INDICE_SOCIALES_CIUDADANAS   968 non-null    float64
 10  INDICE_LECTURA_CRITICA       968 non-null    float64
 11  INDICE_INGLES                968 non-null    float64
 12  INDICE_TOTAL                 968 non-null    float64
 13  COLE_CATEGORIA               968 non-null    object
 14  NOMBRE_DEPARTAMENTO          968 non-null    object
 15  NOMBRE_MUNICIPIO             968 non-null    object
 16  ZONA                         968 non-null    object
 17  LATITUD                      968 non-null    float64
 18  LONGITUD                     968 non-null    float64
 19  MODE_ESTRATOVIVIENDA         968 non-null    object
 20  IND_EST_TRAB                 968 non-null    int64
 21  MODE_EDU_MADRE               968 non-null    object
 22  MODE_EDU_PADRE               968 non-null    object
 23  IND_BILINGUE                 792 non-null    object
 24  CARACTER_IE                  954 non-null    object
dtypes: float64(8), int64(3), object(14)
memory usage: 196.6+ KB
```

5.4.5 ◐ Limpieza Base de datos

In [34]:
```
#Segmentación de variables en numericas y texto
Obj_Col= df.select_dtypes(include='object').columns
numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
Num_df = df.select_dtypes(include=numerics)
```

In [35]:
```
#Limpieza de tildes y 'ñ' del texto para correcta visualización
for i in Obj_Col:
    df[i] = df[i].apply(lambda x: unicodedata.normalize("NFKD", str(x)).encode("ascii","ignore").decode("ascii"))
```

In [36]:
```
#Casteo de variables númericas
numericas = ['INDICE_MATEMATICAS','INDICE_C_NATURALES','INDICE_SOCIALES_CIUDADANAS','INDICE_LECTURA_CRITICA',
             'INDICE_INGLES','INDICE_TOTAL']
df[numericas] = df[numericas].apply(pd.to_numeric)
df.dtypes
```

Out[36]:
```
COD_COL                       int64
COLE_INST_NOMBRE             object
COLE_DEPTO_COLEGIO           object
COLE_CALENDARIO_COLEGIO      object
COLE_GENEROPOBLACION         object
EVALUADOS_ULTIMOS_3           int64
COLE_NATURALEZA              object
INDICE_MATEMATICAS          float64
INDICE_C_NATURALES          float64
INDICE_SOCIALES_CIUDADANAS  float64
INDICE_LECTURA_CRITICA      float64
INDICE_INGLES               float64
INDICE_TOTAL                float64
COLE_CATEGORIA               object
NOMBRE_DEPARTAMENTO          object
NOMBRE_MUNICIPIO             object
ZONA                         object
LATITUD                     float64
LONGITUD                    float64
MODE_ESTRATOVIVIENDA         object
IND_EST_TRAB                  int64
MODE_EDU_MADRE               object
MODE_EDU_PADRE               object
IND_BILINGUE                 object
CARACTER_IE                  object
dtype: object
```

In [37]:
```
#Metrica de ajuste del indice total de las instituciones con el número de estudiantes evaluados
C= df['INDICE_TOTAL'].mean()
m=df['EVALUADOS_ULTIMOS_3'].quantile(.90)
def calculo_metrica (df):
    v= df['EVALUADOS_ULTIMOS_3']
    R= df['INDICE_TOTAL']
    metric = (v/(v+m) *R)+( v/(v+m) *C)
    return metric
```

In [38]:
```
df['INDICE_AJUST']= df.apply(calculo_metrica,axis=1)
df.head(2)
```

Out[38]:

	COD_COL	COLE_INST_NOMBRE	COLE_DEPTO_COLEGIO	COLE_CALENDARIO_COLEGIO	COLE_GENEROPOBLACION	EVALUADOS_ULTIMOS_
0	205790000235	I. E. LA INMACULADA	ANTIOQUIA	A	MI	6
1	105147000568	I. E. JOSE MARIA MUNOZ FLOREZ	ANTIOQUIA	A	MI	26

2 rows × 26 columns

Explicar indice total ----------

In [39]:
```
#Generación de base de datos unificada y limpia para procesamiento posterior (Almacenamiento en S3)
#df.to_csv('BD_IE_DP.csv' , sep= '|')
```

5.5. ◎ Exploracion Bases de Datos

1. Variables Numéricas
   A. INDICE_MATEMATICAS
   B. INDICE_C_NATURALES
   C. INDICE_SOCIALES_CIUDADANAS
   D. INDICE_LECTURA_CRITICA
   E. INDICE_INGLES
   F. INDICE_TOTAL
   G. EVALUADOS_ULTIMOS_3
   H. Correlaciones
2. Variables Categoricas
   A. COLE_CALENDARIO_COLEGIO
   B. COLE_GENEROPOBLACION
   C. COLE_NATURALEZA
   D. COLE_CATEGORIA
   E. ZONA
   F. MODE_ESTRATOVIVIENDA
   G. MODE_EDU_MADRE
   H. MODE_EDU_PADRE
   I. IND_BILINGUE
   J. CARACTER_IE
3. Detección de outliers en sistema de calificación

5.5.1 ◐ Variables Numéricas

1. INDICE_MATEMATICAS
2. INDICE_C_NATURALES
3. INDICE_SOCIALES_CIUDADANAS
4. INDICE_LECTURA_CRITICA
5. INDICE_INGLES
6. INDICE_TOTAL
7. EVALUADOS_ULTIMOS_3
8. Correlaciones

5.5.1.1 ◤ INDICE_MATEMATICAS

In [40]:
```
inter_uncond_descrp_num_var(col = df['INDICE_MATEMATICAS'])
```

Estadísticos

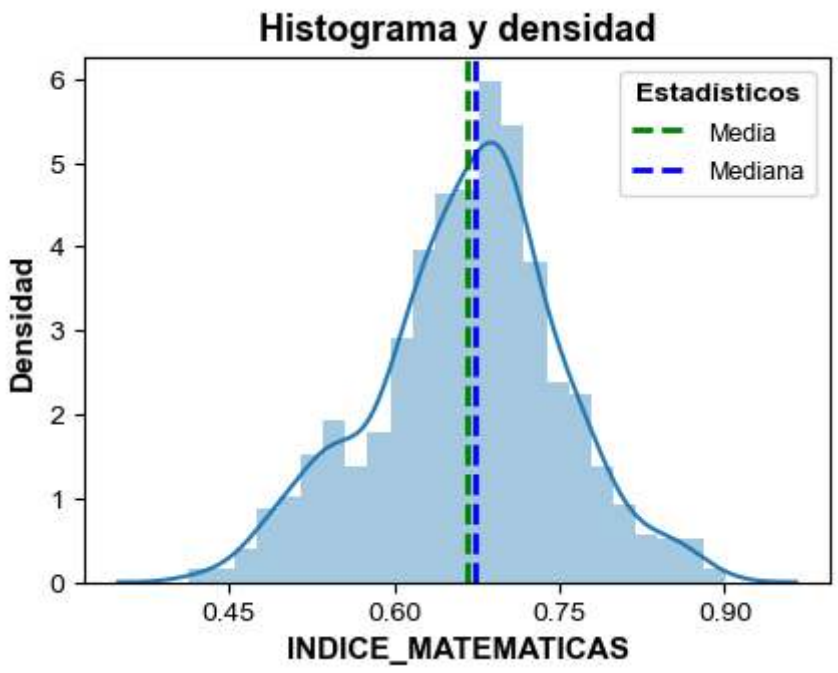
Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Conteo de registros

	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Con dato	968	100.00%	100.00%

Gráficos descriptivos

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [41]:
```
inter_uncond_descrp_num_var(col = df['INDICE_MATEMATICAS'])
```

▾ **Estadísticos**

Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Tendencia central

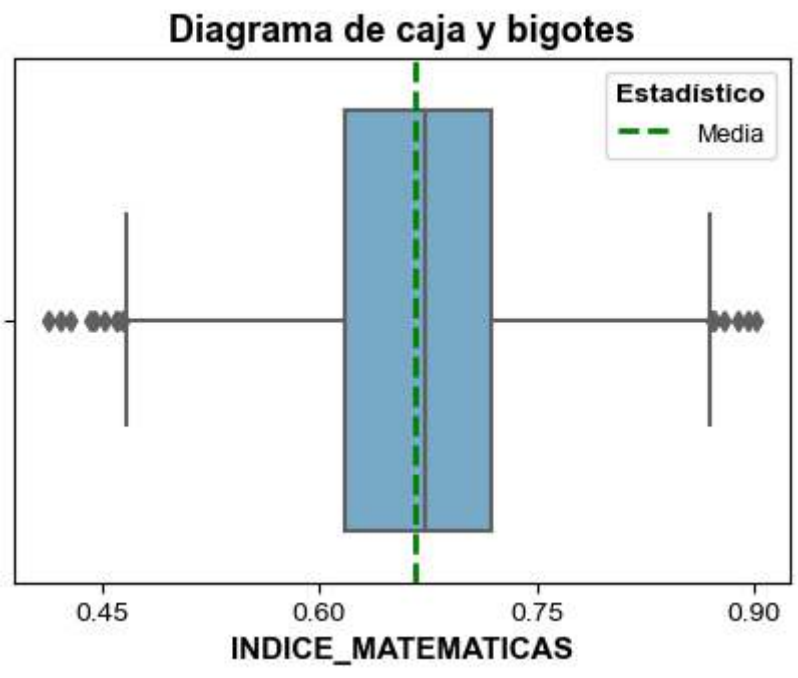
	Resultado
Medida	
Moda	0.68
Media	0.67
Media Armónica	0.66
Media Geométrica	0.66
Media Cuadrática	0.67
Media Trunc.(5%)	0.67
Media IQ	0.67
Media Wins.(5%)	0.67
Trimedia	0.67
Mediana	0.67
Mid Range	0.66
Mid Hinge	0.67

Posición

	Resultado
Medida	
Mínimo	0.41
Percentil 1	0.47
Percentil 5	0.51
Percentil 10	0.55
Percentil 25	0.62
Percentil 50	0.67
Percentil 75	0.72
Percentil 90	0.77
Percentil 95	0.80
Percentil 99	0.86
Máximo	0.90

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [42]:
```
inter_uncond_descrp_num_var(col = df['INDICE_MATEMATICAS'])
```

▾ **Estadísticos**

Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Dispersión

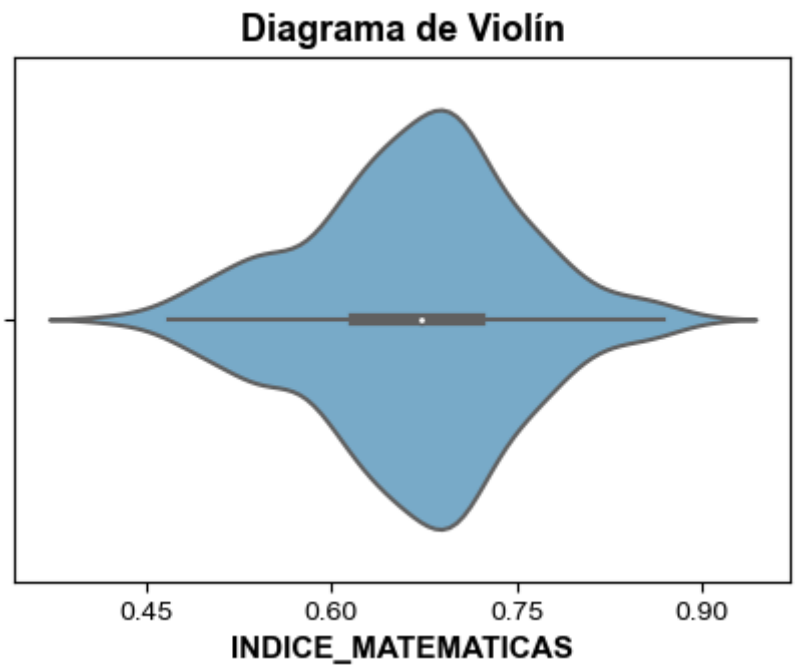
	Resultado
Medida	
Desv. Est.	0.08
Rango	0.49
Rango IQ	0.10
Dif. Abs. Media	0.07
Dif. Abs. Mediana	0.05
Coef. Var.	0.13
QCD	0.08

Forma

	Resultado
Medida	
Asimetría	-0.18
Exc.Curtosis	0.04

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [43]:
```
inter_uncond_descrp_num_var(col = df['INDICE_MATEMATICAS'])
```

▾ **Estadísticos**

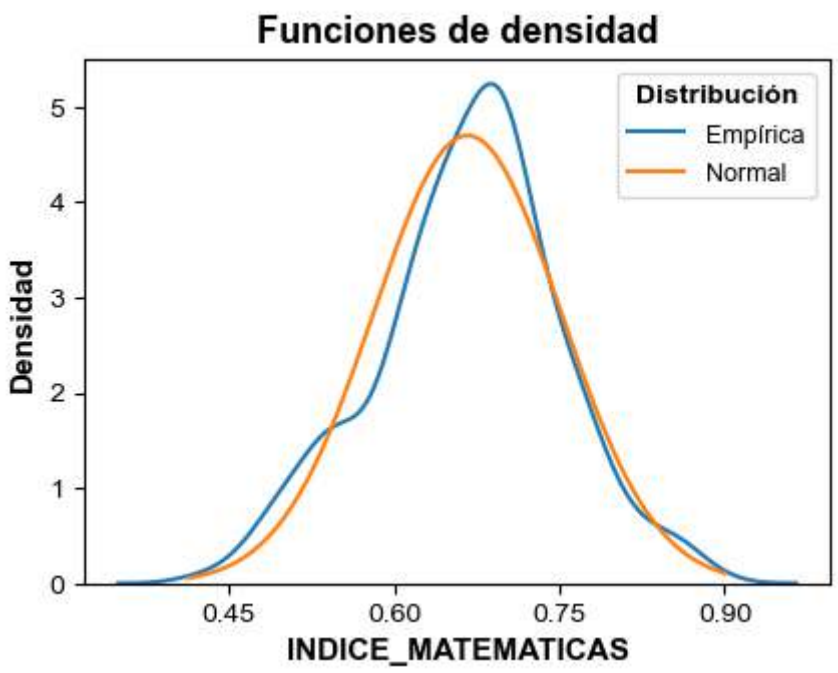
Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Pruebas de normalidad

	Valores P.
Prueba (Test)	
Shapiro-Wilk (SW)	0.01%
Anderson-Darling (AD)	0.00%
Lilliefors (LL)	0.10%
D'Agostino-Pearson (DP)	6.62%
Jarque-Bera (JB)	6.82%
Fisher's Comb. (FC)	0.00%

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.

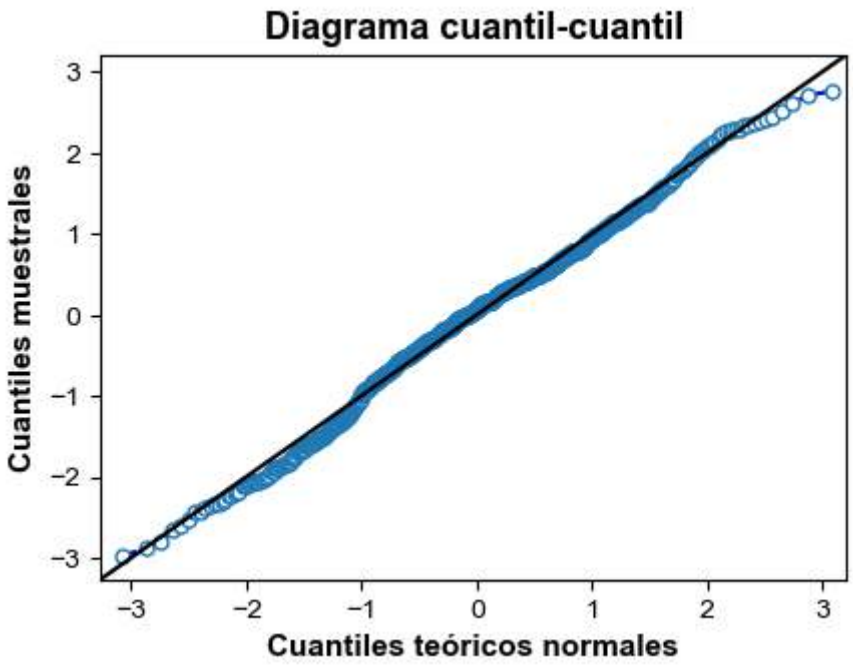


In [44]:
```
inter_uncond_descrp_num_var(col = df['INDICE_MATEMATICAS'])
```

▸ Estadísticos

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



5.5.1.2 INDICE_C_NATURALES

In [45]:
```
inter_uncond_descrp_num_var(col = df['INDICE_C_NATURALES'])
```

▾ **Estadísticos**

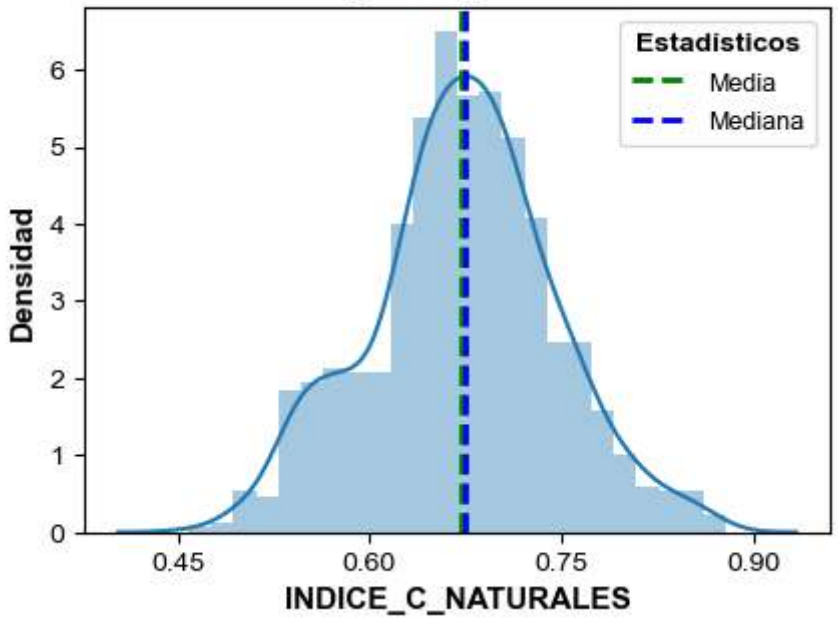
Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Conteo de registros

	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Con dato	968	100.00%	100.00%

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [46]:
```
inter_uncond_descrp_num_var(col = df['INDICE_C_NATURALES'])
```

▾ **Estadísticos**

Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Tendencia central

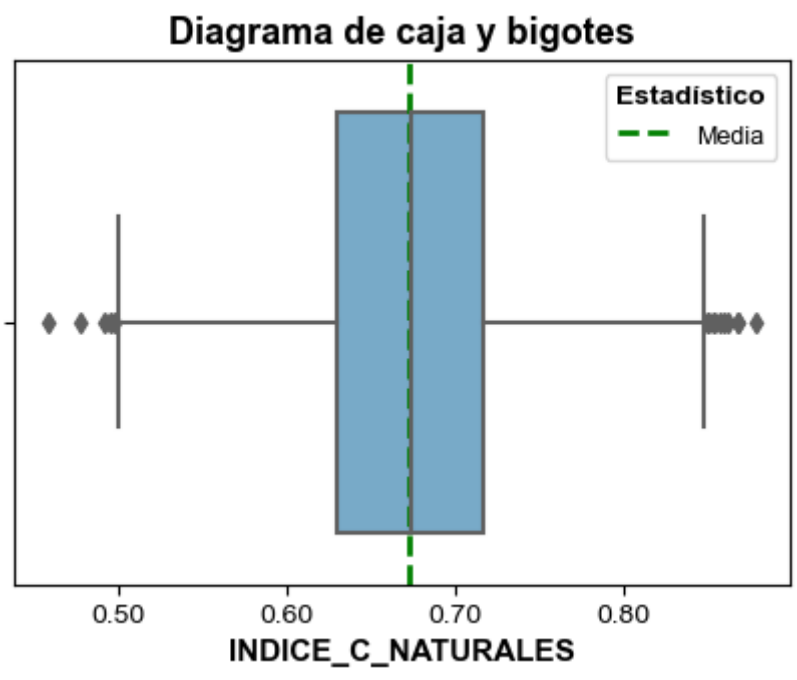
Medida	Resultado
Moda	0.65
Media	0.67
Media Armónica	0.66
Media Geométrica	0.67
Media Cuadrática	0.68
Media Trunc.(5%)	0.67
Media IQ	0.67
Media Wins.(5%)	0.67
Trimedia	0.67
Mediana	0.67
Mid Range	0.67
Mid Hinge	0.67

Posición

Medida	Resultado
Mínimo	0.46
Percentil 1	0.50
Percentil 5	0.54
Percentil 10	0.57
Percentil 25	0.63
Percentil 50	0.67
Percentil 75	0.72
Percentil 90	0.76
Percentil 95	0.79
Percentil 99	0.85
Máximo	0.88

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [47]:
```
inter_uncond_descrp_num_var(col = df['INDICE_C_NATURALES'])
```

▾ **Estadísticos**

Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Dispersión

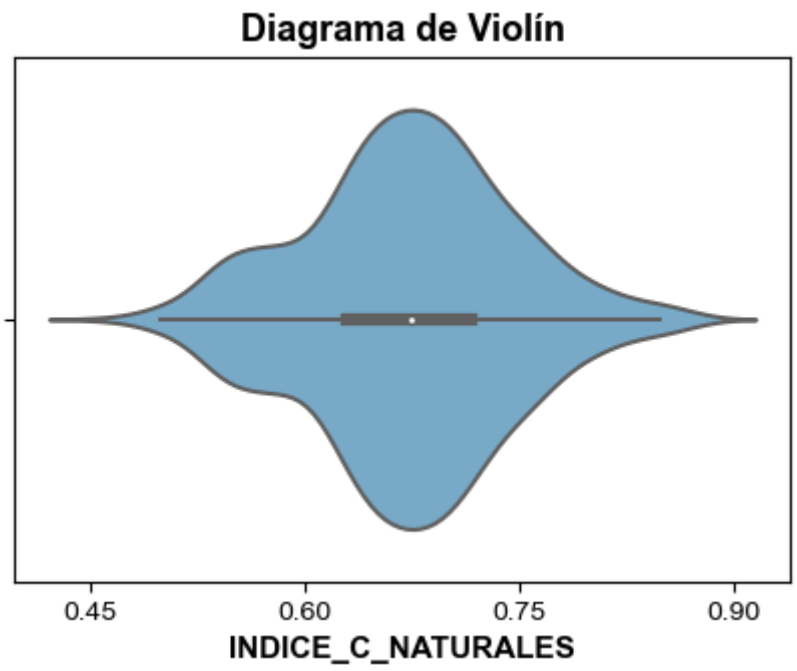
Medida	Resultado
Desv. Est.	0.07
Rango	0.42
Rango IQ	0.09
Dif. Abs. Media	0.06
Dif. Abs. Mediana	0.04
Coef. Var.	0.11
QCD	0.06

Forma

Medida	Resultado
Asimetría	-0.03
Exc.Curtosis	-0.03

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [48]:
```
inter_uncond_descrp_num_var(col = df['INDICE_C_NATURALES'])
```

▾ **Estadísticos**

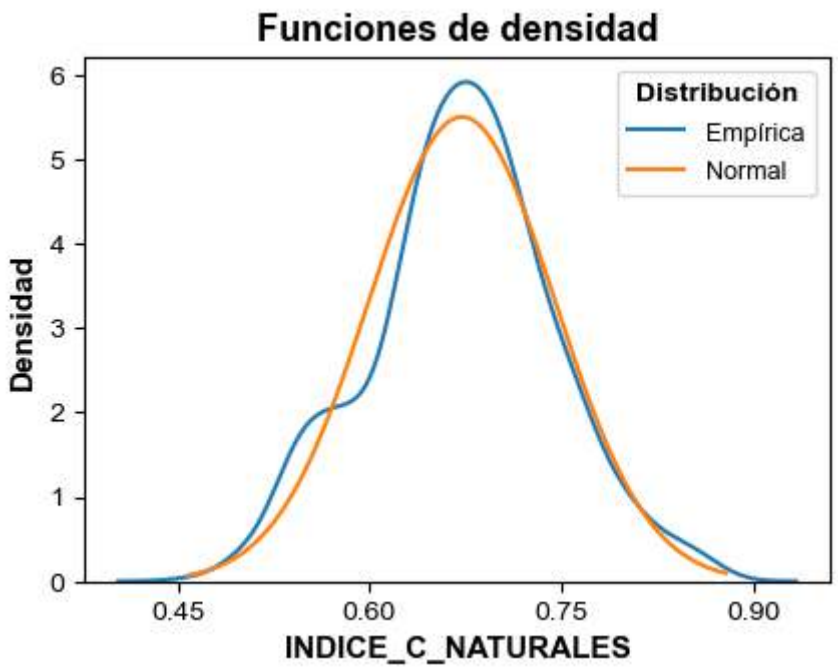
Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Pruebas de normalidad

	Valores P.
Prueba (Test)	
Shapiro-Wilk (SW)	0.21%
Anderson-Darling (AD)	0.03%
Lilliefors (LL)	1.83%
D'Agostino-Pearson (DP)	91.44%
Jarque-Bera (JB)	89.81%
Fisher's Comb. (FC)	0.01%

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.

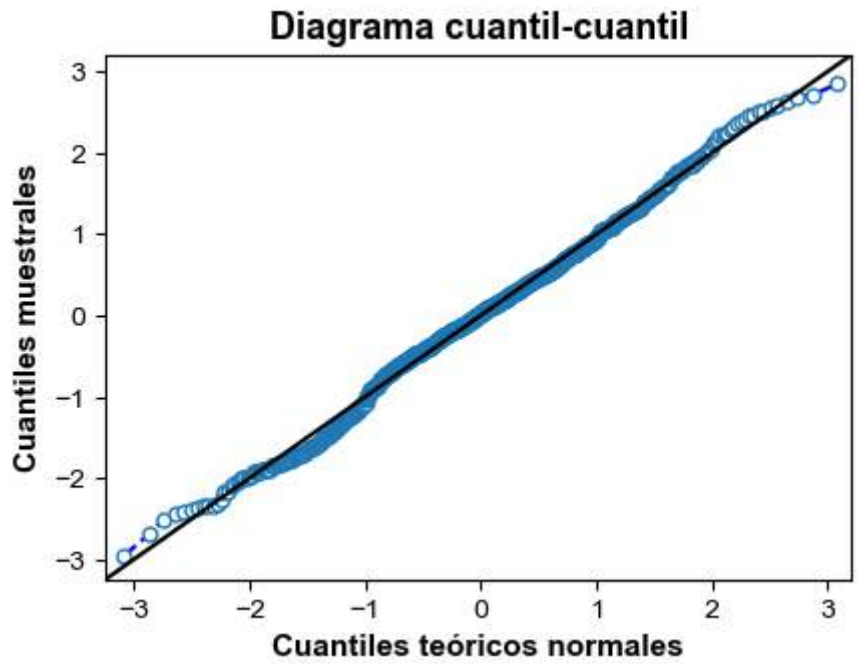


In [49]:
```
inter_uncond_descrp_num_var(col = df['INDICE_C_NATURALES'])
```

▸ Estadísticos

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



5.5.1.3 INDICE_SOCIALES_CIUDADANAS

In [50]:
```
inter_uncond_descrp_num_var(col = df['INDICE_SOCIALES_CIUDADANAS'])
```

Estadísticos

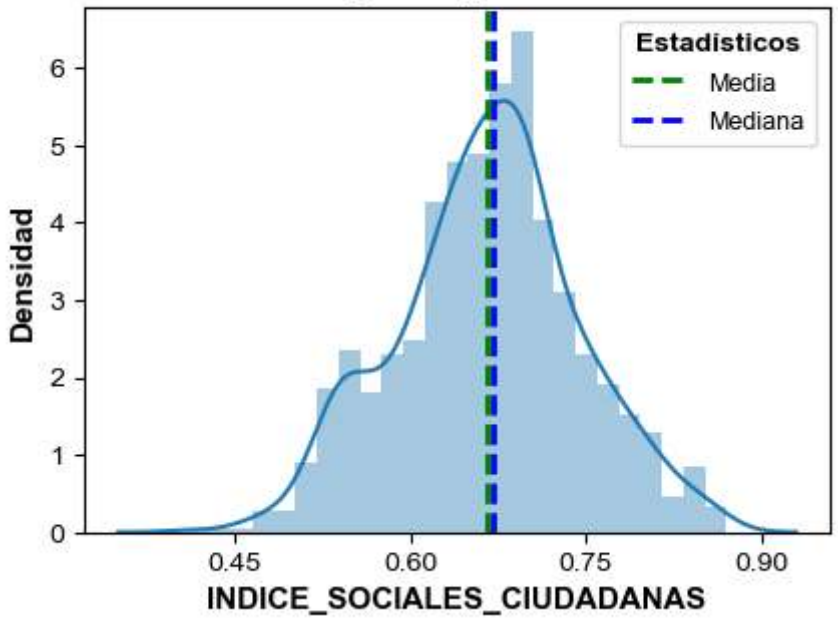
Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Conteo de registros

	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Con dato	968	100.00%	100.00%

Gráficos descriptivos

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [51]:
```
inter_uncond_descrp_num_var(col = df['INDICE_SOCIALES_CIUDADANAS'])
```

▾ **Estadísticos**

Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Tendencia central

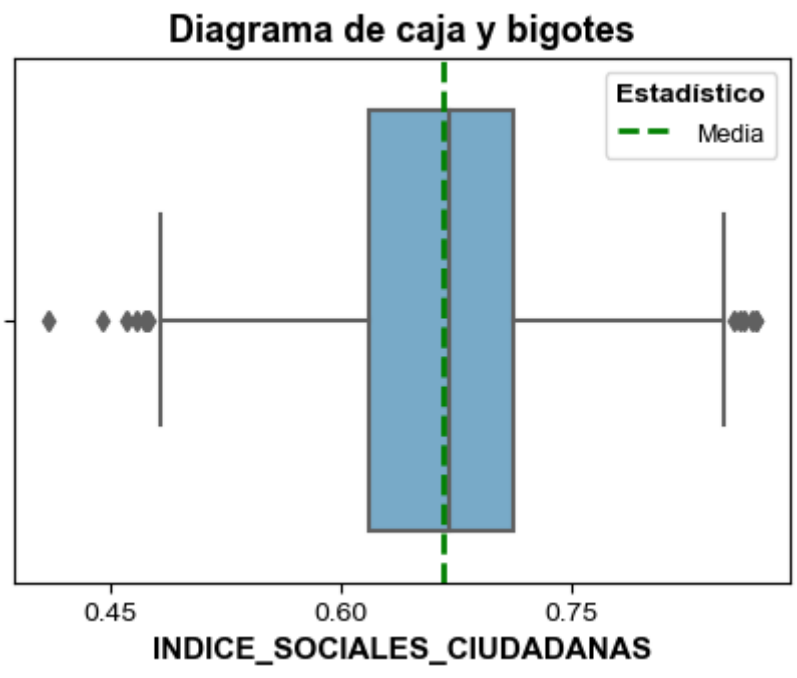
	Resultado
Medida	
Moda	0.63
Media	0.67
Media Armónica	0.66
Media Geométrica	0.66
Media Cuadrática	0.67
Media Trunc.(5%)	0.67
Media IQ	0.67
Media Wins.(5%)	0.67
Trimedia	0.67
Mediana	0.67
Mid Range	0.64
Mid Hinge	0.67

Posición

	Resultado
Medida	
Mínimo	0.41
Percentil 1	0.49
Percentil 5	0.53
Percentil 10	0.55
Percentil 25	0.62
Percentil 50	0.67
Percentil 75	0.71
Percentil 90	0.77
Percentil 95	0.80
Percentil 99	0.85
Máximo	0.87

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [52]:
```
inter_uncond_descrp_num_var(col = df['INDICE_SOCIALES_CIUDADANAS'])
```

▾ **Estadísticos**

Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Dispersión

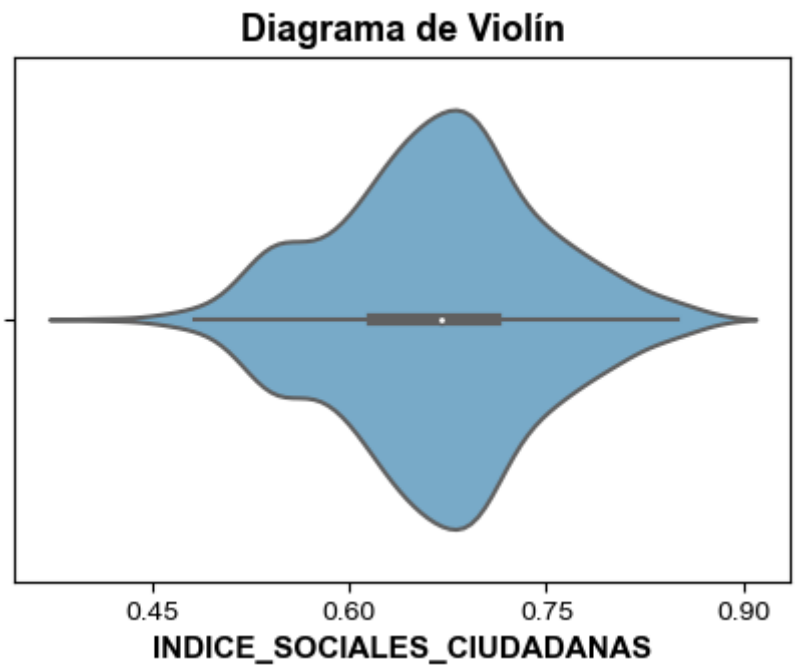
	Resultado
Medida	
Desv. Est.	0.08
Rango	0.46
Rango IQ	0.09
Dif. Abs. Media	0.06
Dif. Abs. Mediana	0.05
Coef. Var.	0.12
QCD	0.07

Forma

	Resultado
Medida	
Asimetría	-0.05
Exc.Curtosis	-0.11

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [53]:
```
inter_uncond_descrp_num_var(col = df['INDICE_SOCIALES_CIUDADANAS'])
```

▾ **Estadísticos**

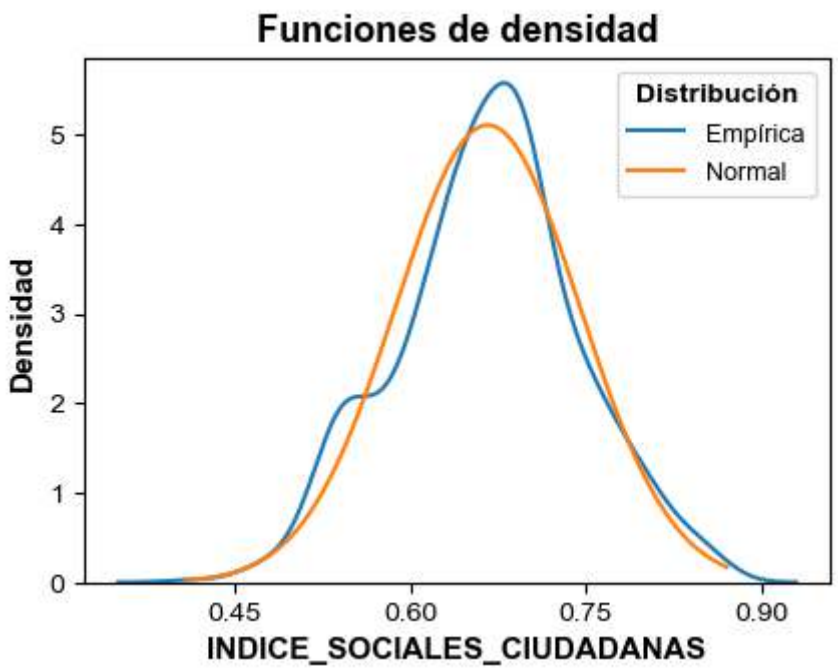
Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Pruebas de normalidad

	Valores P.
Prueba (Test)	
Shapiro-Wilk (SW)	0.30%
Anderson-Darling (AD)	0.04%
Lilliefors (LL)	3.59%
D'Agostino-Pearson (DP)	63.55%
Jarque-Bera (JB)	61.01%
Fisher's Comb. (FC)	0.01%

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.

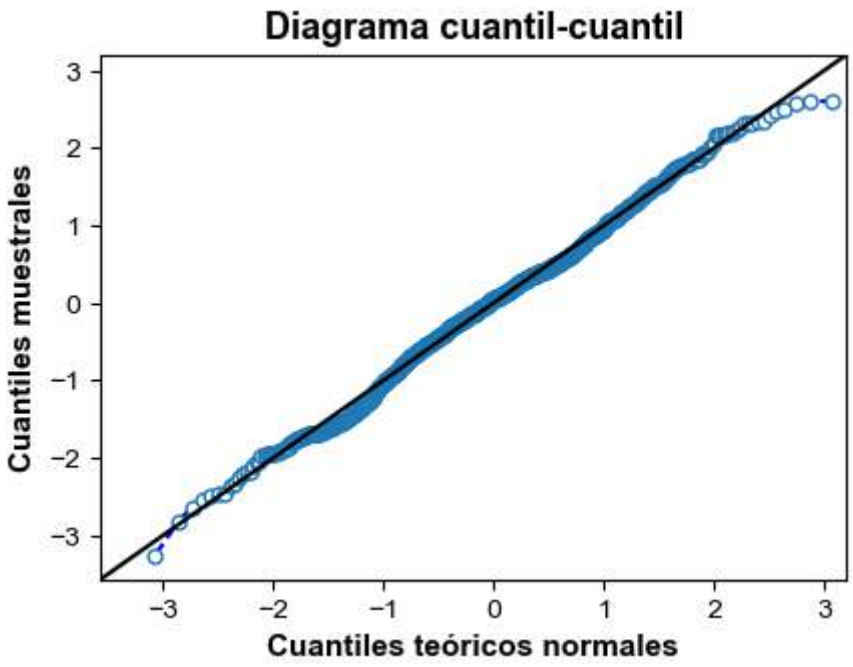


In [54]:
```
inter_uncond_descrp_num_var(col = df['INDICE_SOCIALES_CIUDADANAS'])
```

▸ Estadísticos

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



5.5.1.4 INDICE_LECTURA_CRITICA

In [55]:
```
inter_uncond_descrp_num_var(col = df['INDICE_LECTURA_CRITICA'])
```

Estadísticos

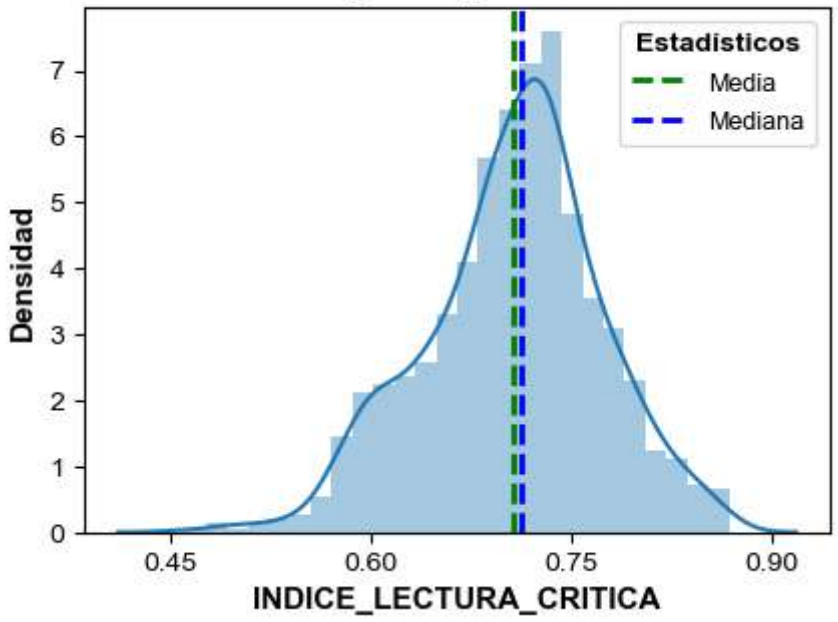
Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Conteo de registros

	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Con dato	968	100.00%	100.00%

Gráficos descriptivos

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [56]:
```
inter_uncond_descrp_num_var(col = df['INDICE_LECTURA_CRITICA'])
```

▾ **Estadísticos**

Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Tendencia central

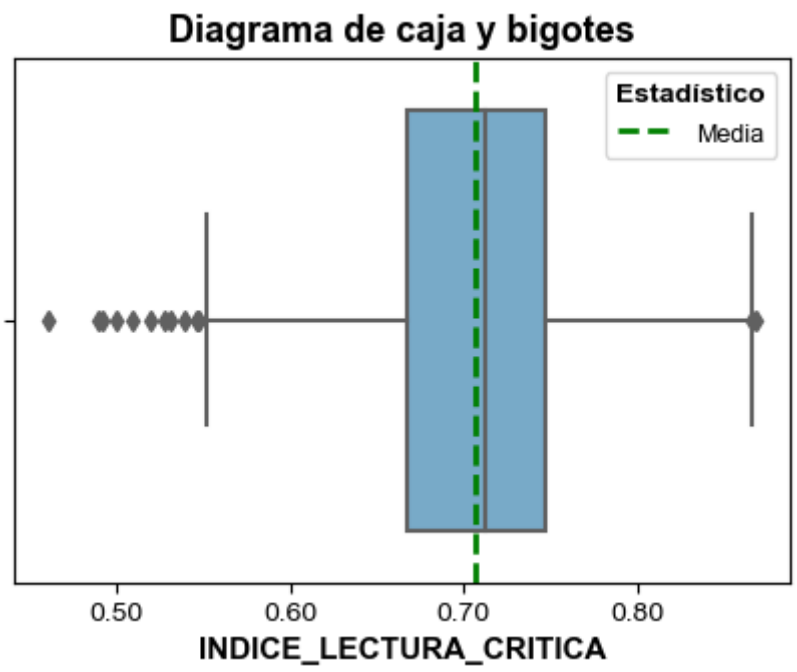
	Resultado
Medida	
Moda	0.72
Media	0.71
Media Armónica	0.70
Media Geométrica	0.70
Media Cuadrática	0.71
Media Trunc.(5%)	0.71
Media IQ	0.71
Media Wins.(5%)	0.71
Trimedia	0.71
Mediana	0.71
Mid Range	0.66
Mid Hinge	0.71

Posición

	Resultado
Medida	
Mínimo	0.46
Percentil 1	0.55
Percentil 5	0.59
Percentil 10	0.61
Percentil 25	0.67
Percentil 50	0.71
Percentil 75	0.75
Percentil 90	0.79
Percentil 95	0.81
Percentil 99	0.85
Máximo	0.87

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [57]:
```
inter_uncond_descrp_num_var(col = df['INDICE_LECTURA_CRITICA'])
```

▾ **Estadísticos**

Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Dispersión

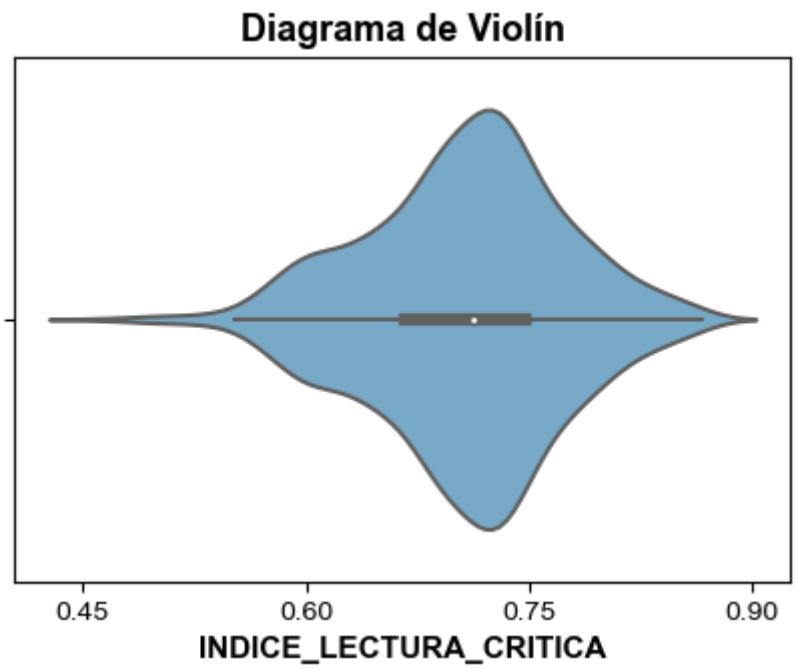
	Resultado
Medida	
Desv. Est.	0.07
Rango	0.41
Rango IQ	0.08
Dif. Abs. Media	0.05
Dif. Abs. Mediana	0.04
Coef. Var.	0.09
QCD	0.06

Forma

	Resultado
Medida	
Asimetría	-0.30
Exc.Curtosis	0.14

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [58]:
```
inter_uncond_descrp_num_var(col = df['INDICE_LECTURA_CRITICA'])
```

▾ **Estadísticos**

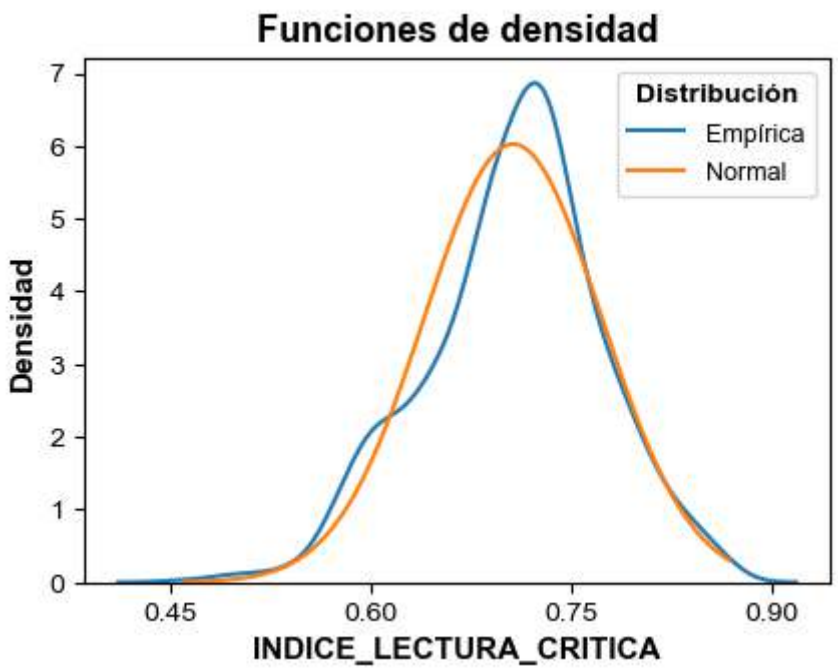
Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Pruebas de normalidad

	Valores P.
Prueba (Test)	
Shapiro-Wilk (SW)	0.00%
Anderson-Darling (AD)	0.00%
Lilliefors (LL)	0.10%
D'Agostino-Pearson (DP)	0.06%
Jarque-Bera (JB)	0.05%
Fisher's Comb. (FC)	0.00%

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.

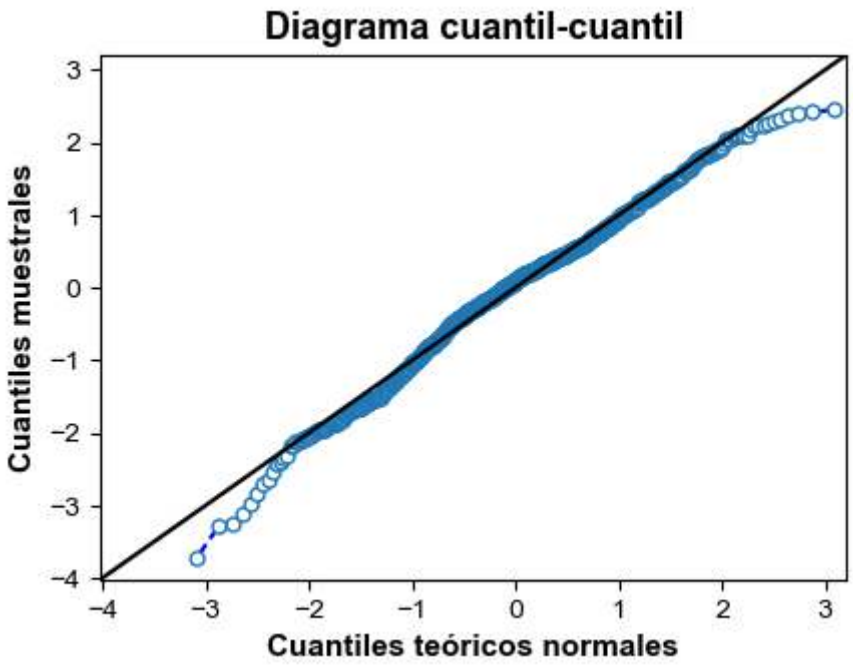


In [59]:
```
inter_uncond_descrp_num_var(col = df['INDICE_LECTURA_CRITICA'])
```

▸ Estadísticos

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



5.5.1.5 INDICE_INGLES

In [60]:
```
inter_uncond_descrp_num_var(col = df['INDICE_INGLES'])
```

▾ **Estadísticos**

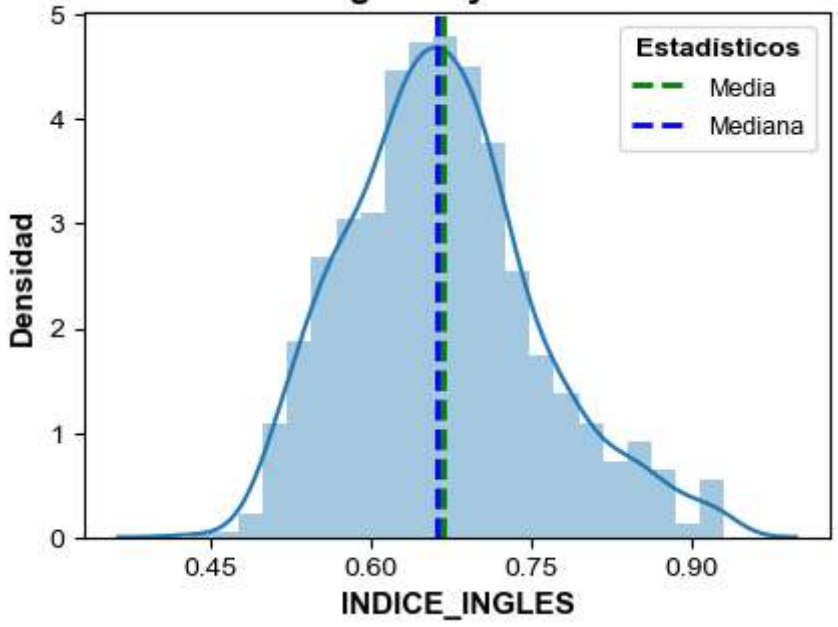
Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Conteo de registros

	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Con dato	968	100.00%	100.00%

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [61]:
```
inter_uncond_descrp_num_var(col = df['INDICE_INGLES'])
```

▾ **Estadísticos**

Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Tendencia central

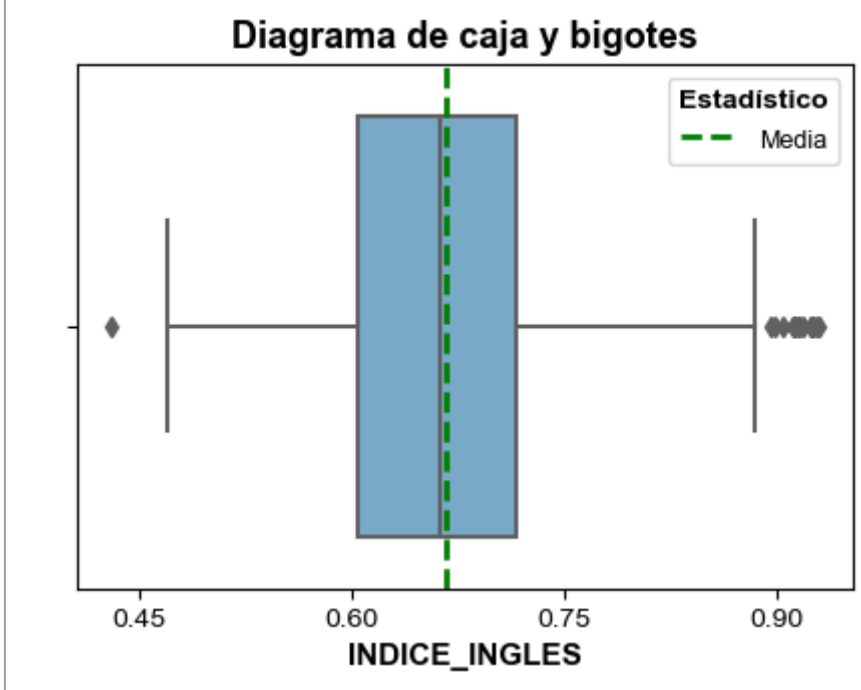
	Resultado
Medida	
Moda	0.57
Media	0.67
Media Armónica	0.66
Media Geométrica	0.66
Media Cuadrática	0.67
Media Trunc.(5%)	0.66
Media IQ	0.66
Media Wins.(5%)	0.67
Trimedia	0.66
Mediana	0.66
Mid Range	0.68
Mid Hinge	0.66

Posición

	Resultado
Medida	
Mínimo	0.43
Percentil 1	0.50
Percentil 5	0.53
Percentil 10	0.56
Percentil 25	0.60
Percentil 50	0.66
Percentil 75	0.72
Percentil 90	0.78
Percentil 95	0.84
Percentil 99	0.91
Máximo	0.93

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [62]:
```
inter_uncond_descrp_num_var(col = df['INDICE_INGLES'])
```

▾ **Estadísticos**

Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Dispersión

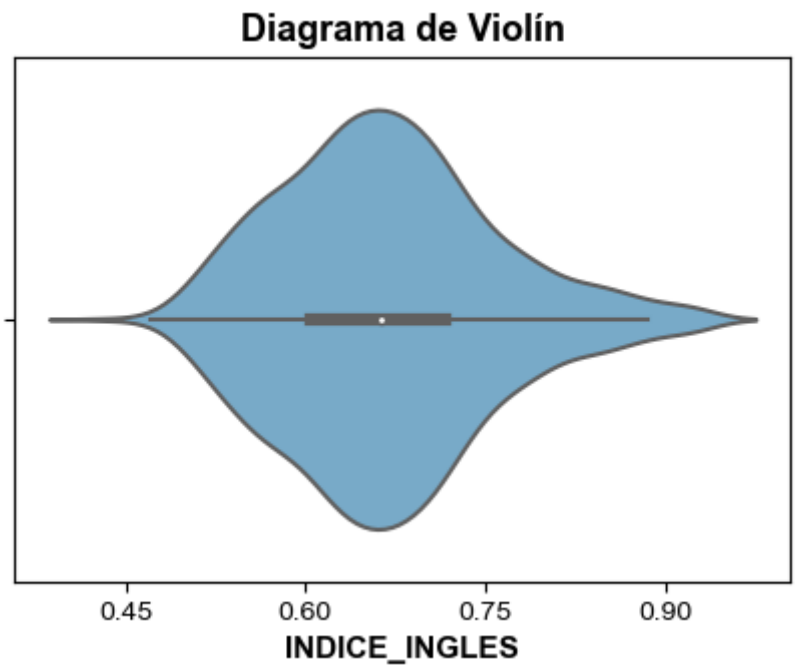
Medida	Resultado
Desv. Est.	0.09
Rango	0.50
Rango IQ	0.11
Dif. Abs. Media	0.07
Dif. Abs. Mediana	0.06
Coef. Var.	0.13
QCD	0.09

Forma

Medida	Resultado
Asimetría	0.49
Exc.Curtosis	0.14

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [63]:
```
inter_uncond_descrp_num_var(col = df['INDICE_INGLES'])
```

▾ **Estadísticos**

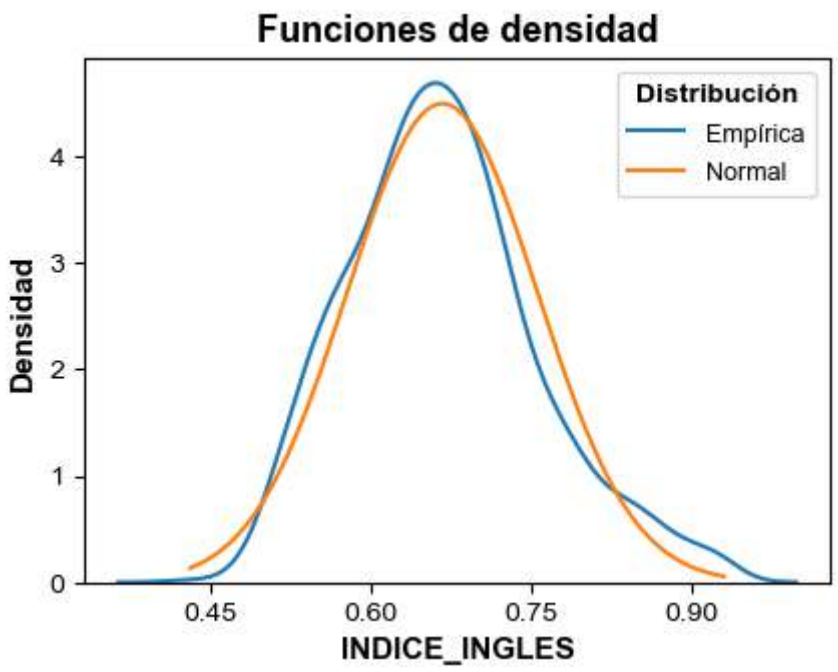
Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Pruebas de normalidad

	Valores P.
Prueba (Test)	
Shapiro-Wilk (SW)	0.00%
Anderson-Darling (AD)	0.00%
Lilliefors (LL)	0.10%
D'Agostino-Pearson (DP)	0.00%
Jarque-Bera (JB)	0.00%
Fisher's Comb. (FC)	0.00%

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.

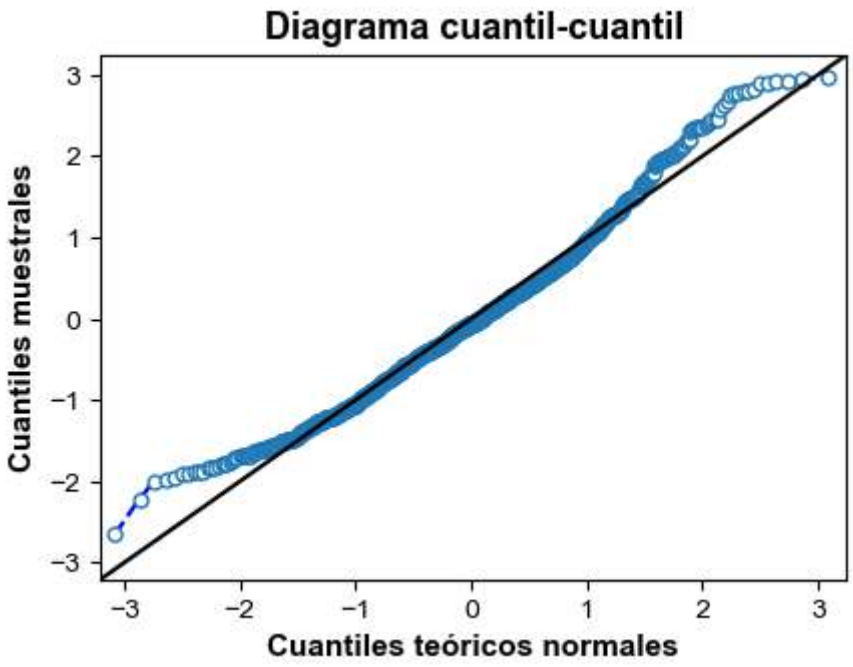


In [64]:
```
inter_uncond_descrp_num_var(col = df['INDICE_INGLES'])
```

▸ Estadísticos

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



5.5.1.6 INDICE_TOTAL

In [65]:
```
inter_uncond_descrp_num_var(col = df['INDICE_TOTAL'])
```

Estadísticos

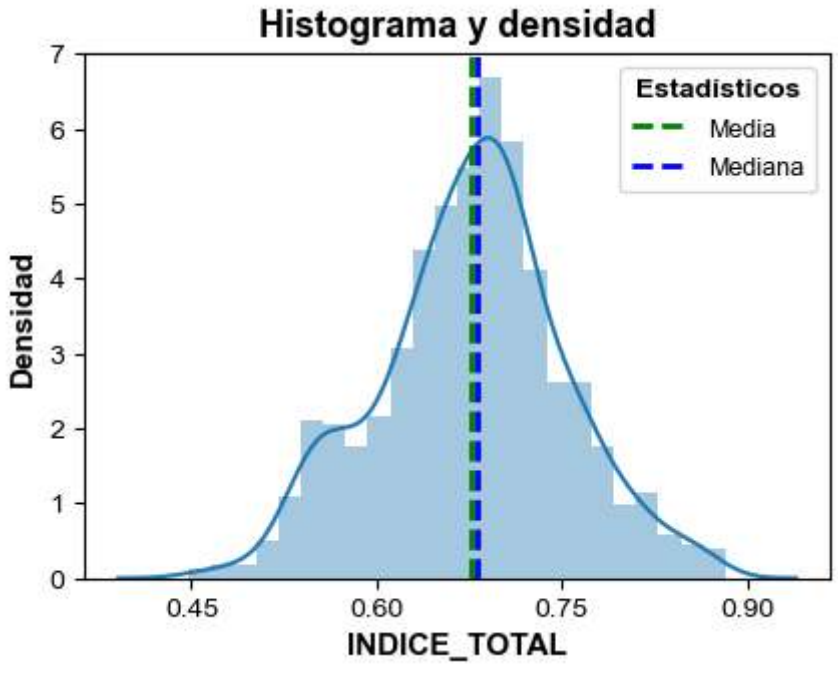
Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Conteo de registros

	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Con dato	968	100.00%	100.00%

Gráficos descriptivos

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [66]:
```
inter_uncond_descrp_num_var(col = df['INDICE_TOTAL'])
```

▾ **Estadísticos**

Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Tendencia central

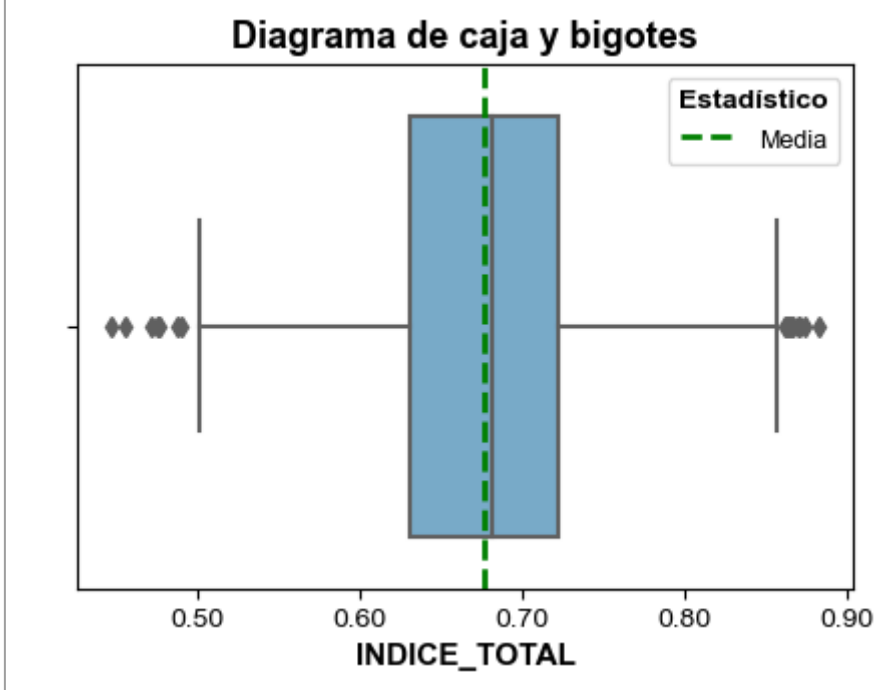
	Resultado
Medida	
Moda	0.69
Media	0.68
Media Armónica	0.67
Media Geométrica	0.67
Media Cuadrática	0.68
Media Trunc.(5%)	0.68
Media IQ	0.68
Media Wins.(5%)	0.68
Trimedia	0.68
Mediana	0.68
Mid Range	0.67
Mid Hinge	0.68

Posición

	Resultado
Medida	
Mínimo	0.45
Percentil 1	0.51
Percentil 5	0.55
Percentil 10	0.57
Percentil 25	0.63
Percentil 50	0.68
Percentil 75	0.72
Percentil 90	0.77
Percentil 95	0.80
Percentil 99	0.86
Máximo	0.88

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [67]:
```
inter_uncond_descrp_num_var(col = df['INDICE_TOTAL'])
```

▾ **Estadísticos**

Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Dispersión

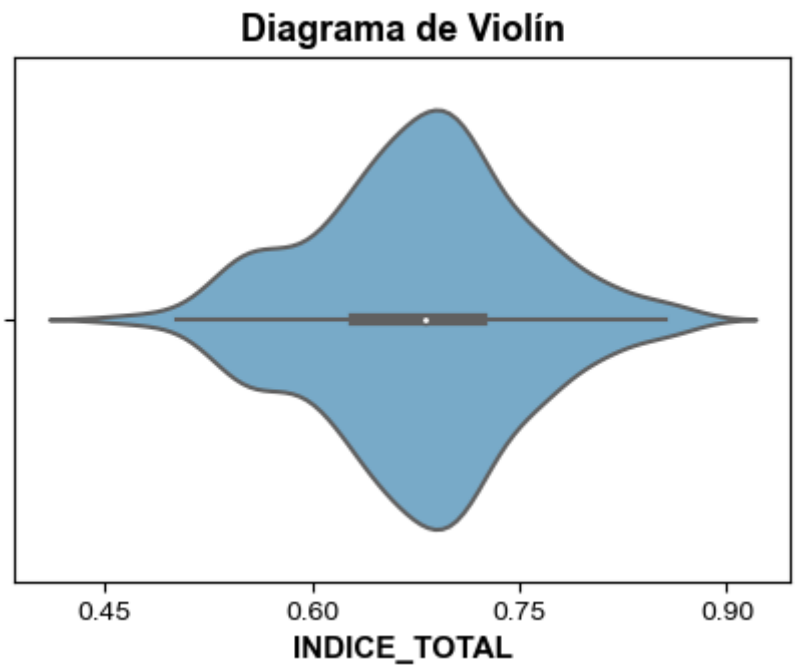
	Resultado
Medida	
Desv. Est.	0.08
Rango	0.43
Rango IQ	0.09
Dif. Abs. Media	0.06
Dif. Abs. Mediana	0.04
Coef. Var.	0.11
QCD	0.07

Forma

	Resultado
Medida	
Asimetría	-0.07
Exc.Curtosis	-0.02

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [68]:
```
inter_uncond_descrp_num_var(col = df['INDICE_TOTAL'])
```

▾ **Estadísticos**

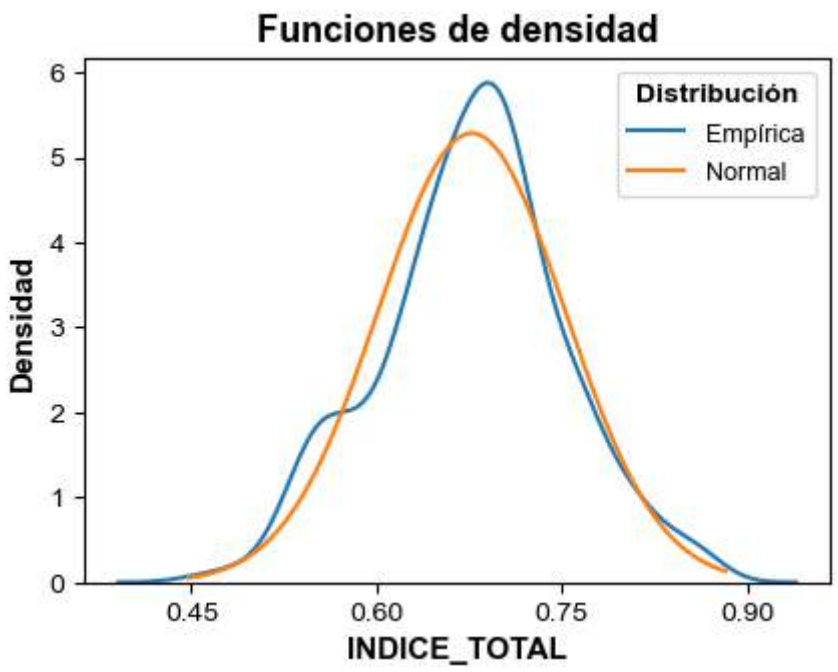
Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Pruebas de normalidad

	Valores P.
Prueba (Test)	
Shapiro-Wilk (SW)	0.12%
Anderson-Darling (AD)	0.01%
Lilliefors (LL)	0.54%
D'Agostino-Pearson (DP)	66.45%
Jarque-Bera (JB)	65.78%
Fisher's Comb. (FC)	0.00%

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.

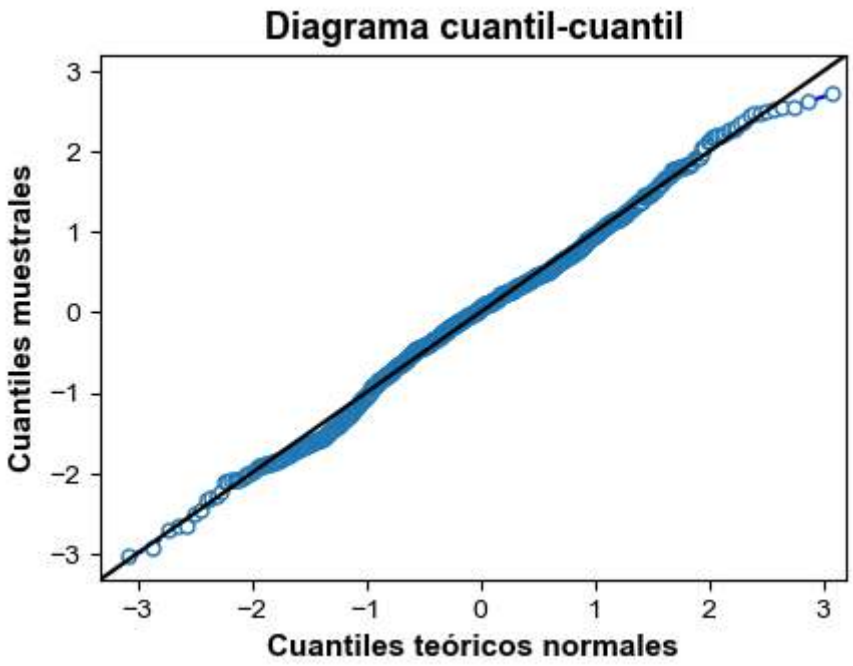


In [69]:
```
inter_uncond_descrp_num_var(col = df['INDICE_TOTAL'])
```

▸ Estadísticos

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



5.5.1.7 EVALUADOS_ULTIMOS_3

In [70]:
```
inter_uncond_descrp_num_var(col = df['EVALUADOS_ULTIMOS_3'])
```

▾ **Estadísticos**

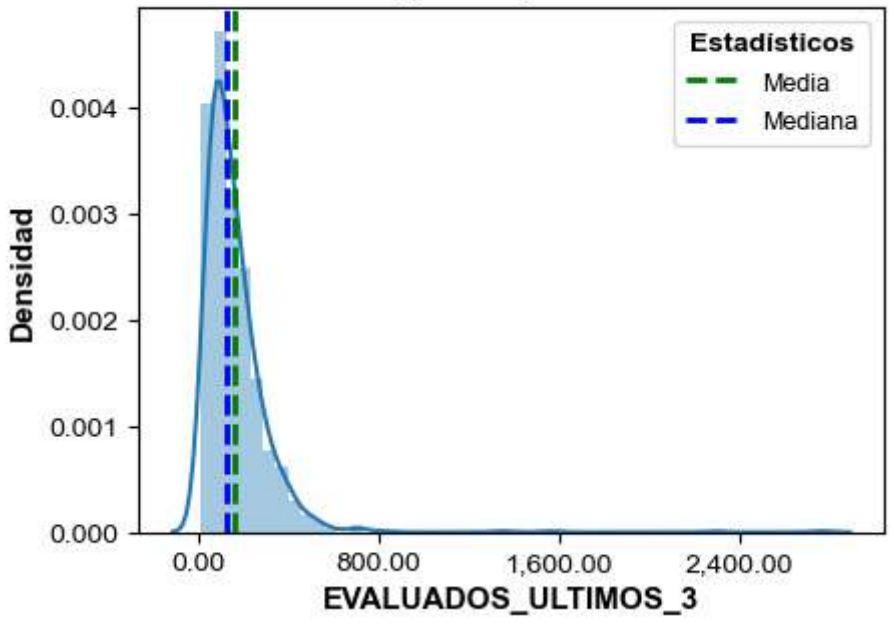
Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Conteo de registros

	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Con dato	968	100.00%	100.00%

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [71]:
```
inter_uncond_descrp_num_var(col = df['EVALUADOS_ULTIMOS_3'])
```

▾ **Estadísticos**

Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Tendencia central

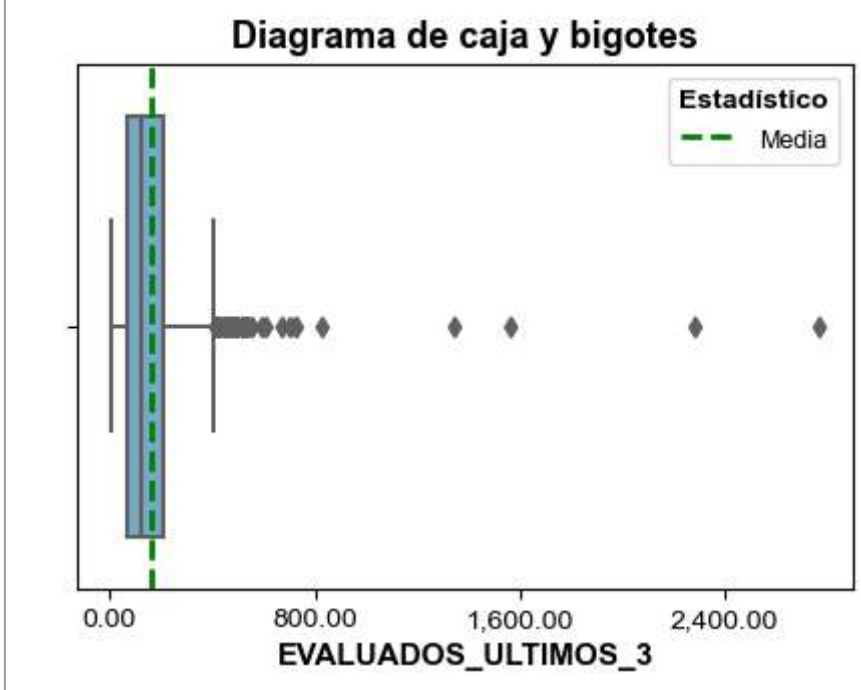
	Resultado
Medida	
Moda	59.00
Media	160.46
Media Armónica	85.51
Media Geométrica	118.20
Media Cuadrática	232.06
Media Trunc.(5%)	143.29
Media IQ	128.23
Media Wins.(5%)	149.80
Trimedia	131.00
Mediana	124.00
Mid Range	1,388.00
Mid Hinge	138.00

Posición

	Resultado
Medida	
Mínimo	9.00
Percentil 1	17.00
Percentil 5	32.00
Percentil 10	44.00
Percentil 25	70.00
Percentil 50	124.00
Percentil 75	206.00
Percentil 90	308.30
Percentil 95	385.30
Percentil 99	596.62
Máximo	2,767.00

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [72]:
```
inter_uncond_descrp_num_var(col = df['EVALUADOS_ULTIMOS_3'])
```

▾ **Estadísticos**

Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Dispersión

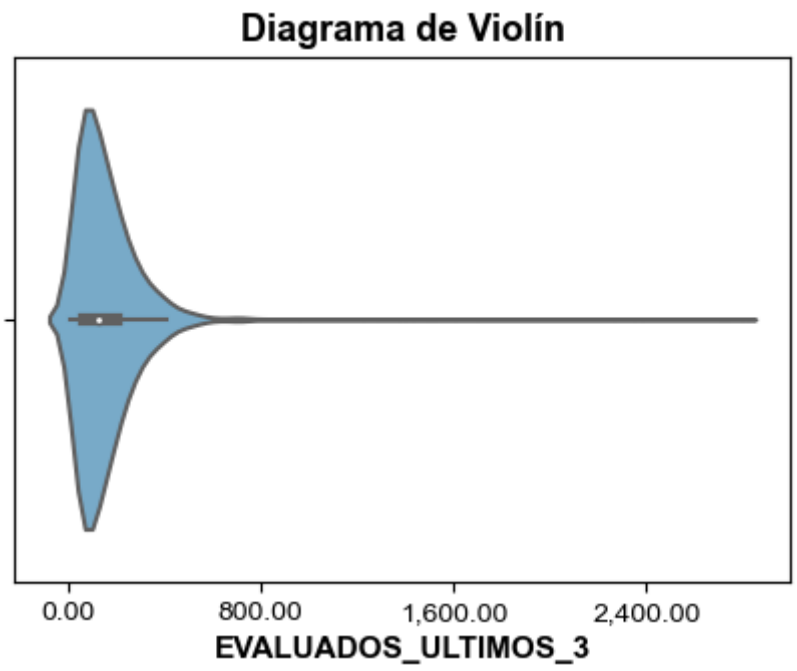
	Resultado
Medida	
Desv. Est.	167.73
Rango	2,758.00
Rango IQ	136.00
Dif. Abs. Media	94.84
Dif. Abs. Mediana	63.00
Coef. Var.	1.04
QCD	0.49

Forma

	Resultado
Medida	
Asimetría	7.41
Exc.Curtosis	93.39

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.



In [73]:
```
inter_uncond_descrp_num_var(col = df['EVALUADOS_ULTIMOS_3'])
```

▾ **Estadísticos**

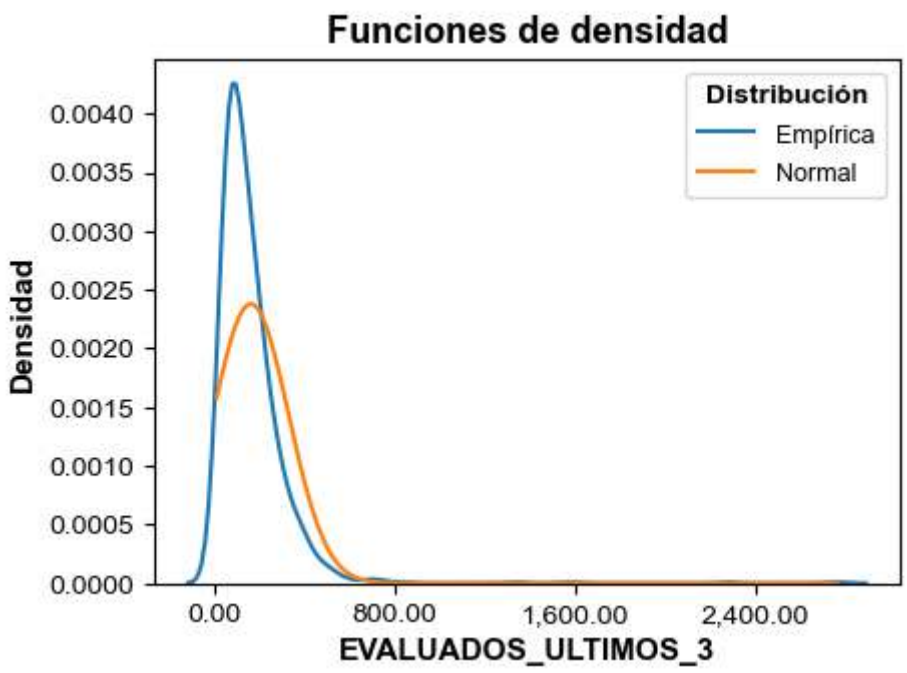
Frecuencia | TC y Posición | Dispersión y Forma | Normalidad

Pruebas de normalidad

	Valores P.
Prueba (Test)	
Shapiro-Wilk (SW)	0.00%
Anderson-Darling (AD)	0.00%
Lilliefors (LL)	0.10%
D'Agostino-Pearson (DP)	0.00%
Jarque-Bera (JB)	0.00%
Fisher's Comb. (FC)	0.00%

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.

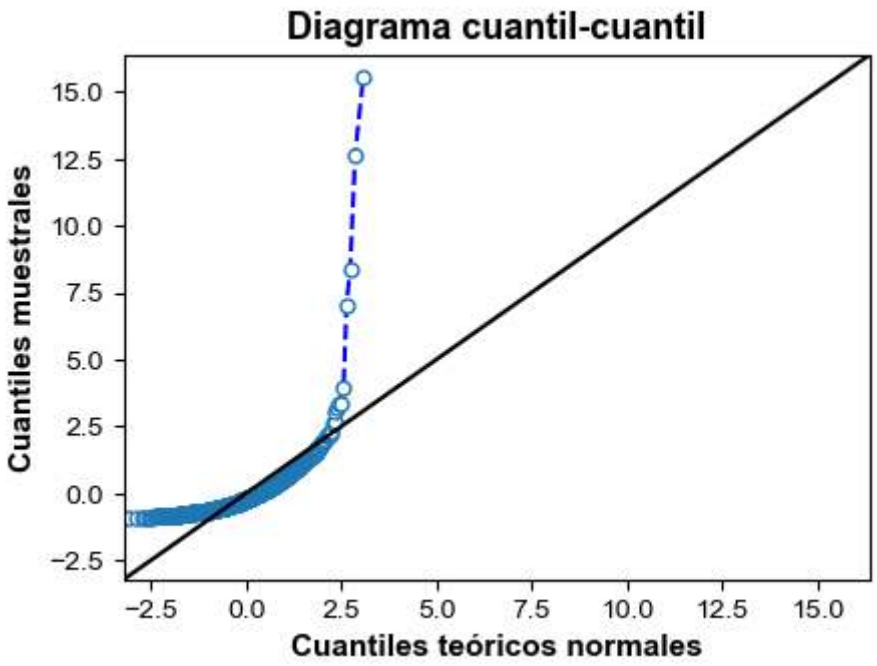


In [74]:
```
inter_uncond_descrp_num_var(col = df['EVALUADOS_ULTIMOS_3'])
```

▸ Estadísticos

▾ **Gráficos descriptivos**

Histograma | Caja y bigotes | Violín | Densidad Norm. | QQ Norm.

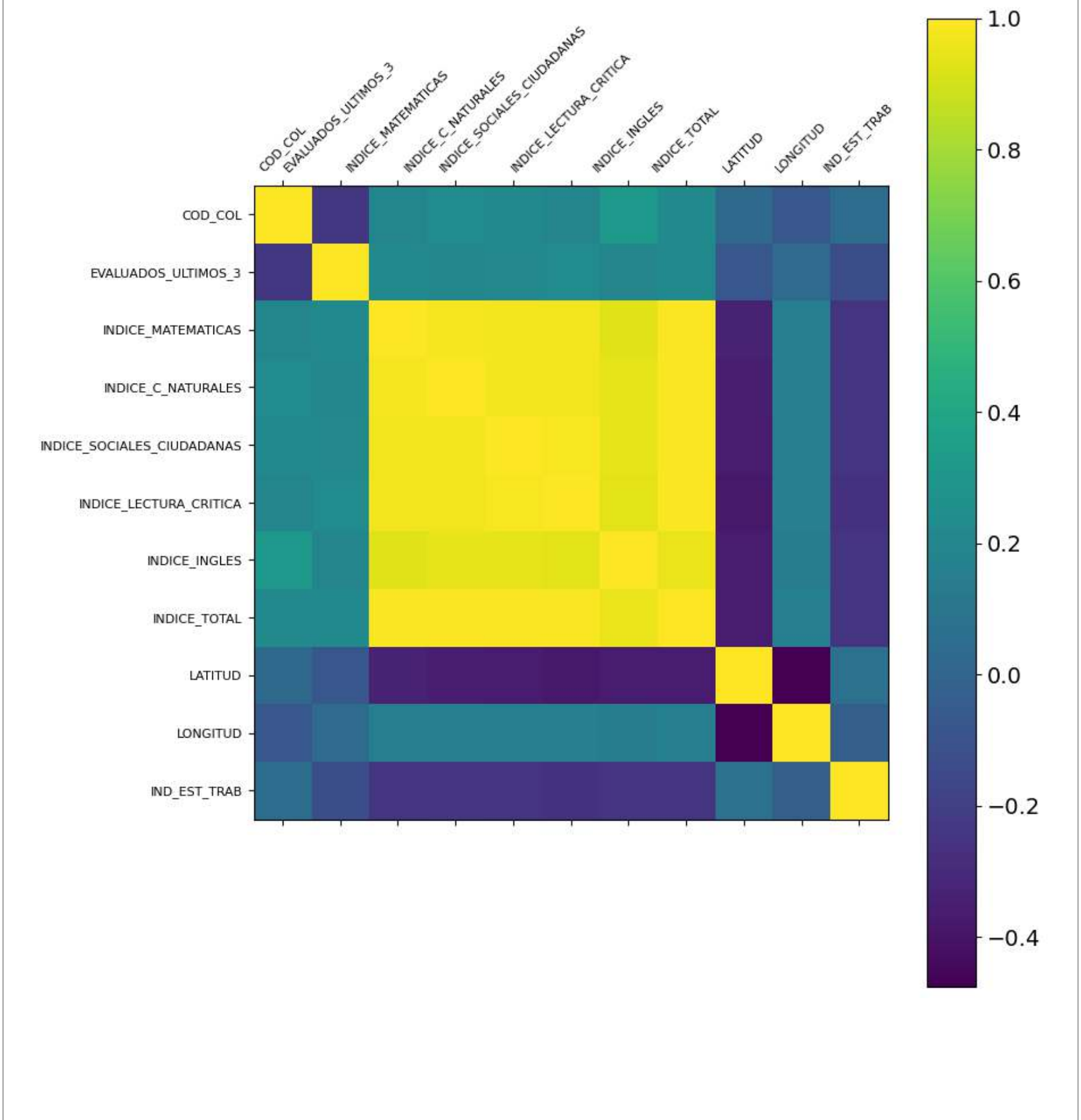


5.5.1.8 Correlaciones

In [75]:
```
print_corr_var(Num_df)
```

Matriz correlación

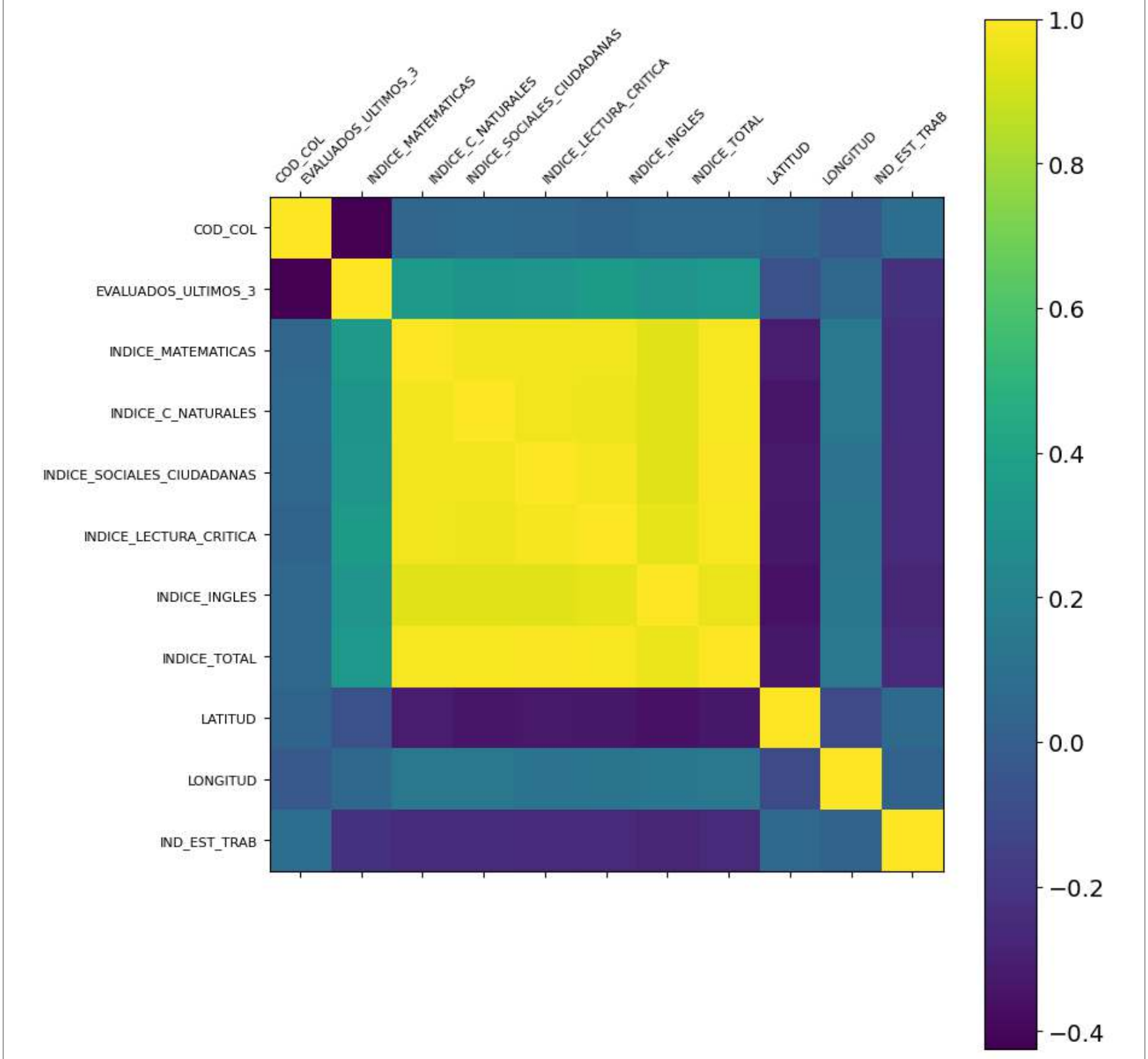
Pearson | Spearman | Kendall



In [76]:
```
print_corr_var(Num_df)
```

Matriz correlación

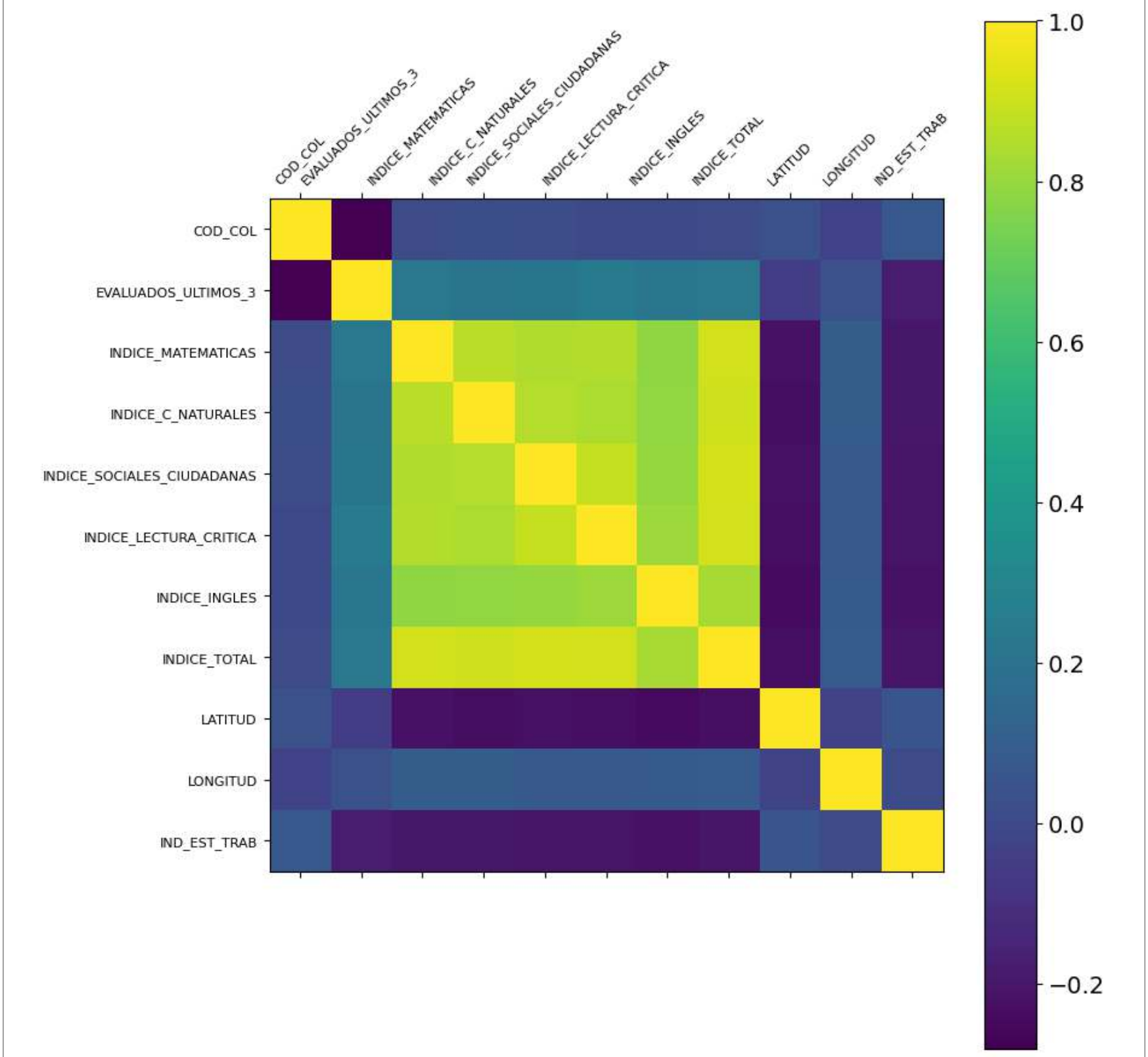
Pearson | Spearman | Kendall



In [77]:
```
print_corr_var(Num_df)
```

Matriz correlación

Pearson | Spearman | Kendall



5.5.2 ◐ Variables Categoricas

1. COLE_CALENDARIO_COLEGIO
2. COLE_GENEROPOBLACION

3. COLE_NATURALEZA
4. COLE_CATEGORIA
5. ZONA
6. MODE_ESTRATOVIVIENDA
7. MODE_EDU_MADRE
8. MODE_EDU_PADRE
9. IND_BILINGUE
10. CARACTER_IE

5.5.2.1 COLE_CALENDARIO_COLEGIO

In [78]:
```
inter_uncond_descrp_cat_var(col = df['COLE_CALENDARIO_COLEGIO'])
```

Estadísticos

Frecuencia

Conteo de registros

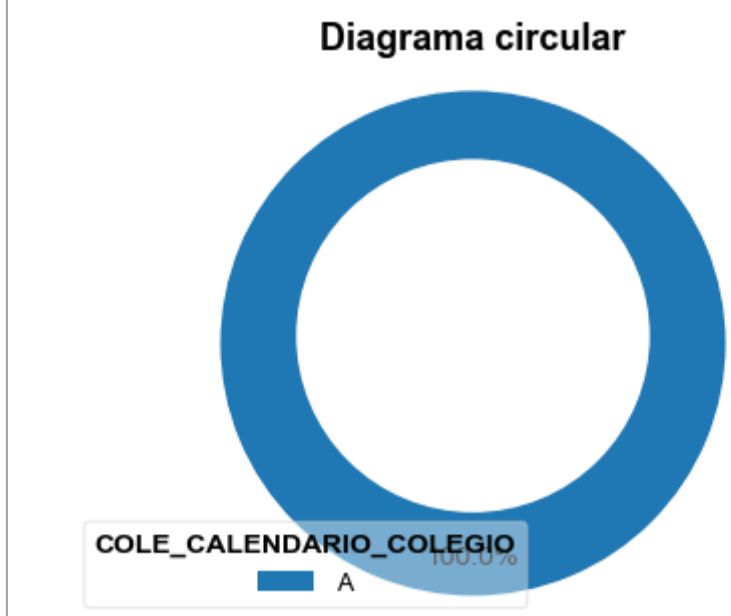
	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Con dato	968	100.00%	100.00%

Conteo de frecuencias

	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Categorías			
A	968	100.00%	100.00%

Gráficos descriptivos

Circular | Pareto

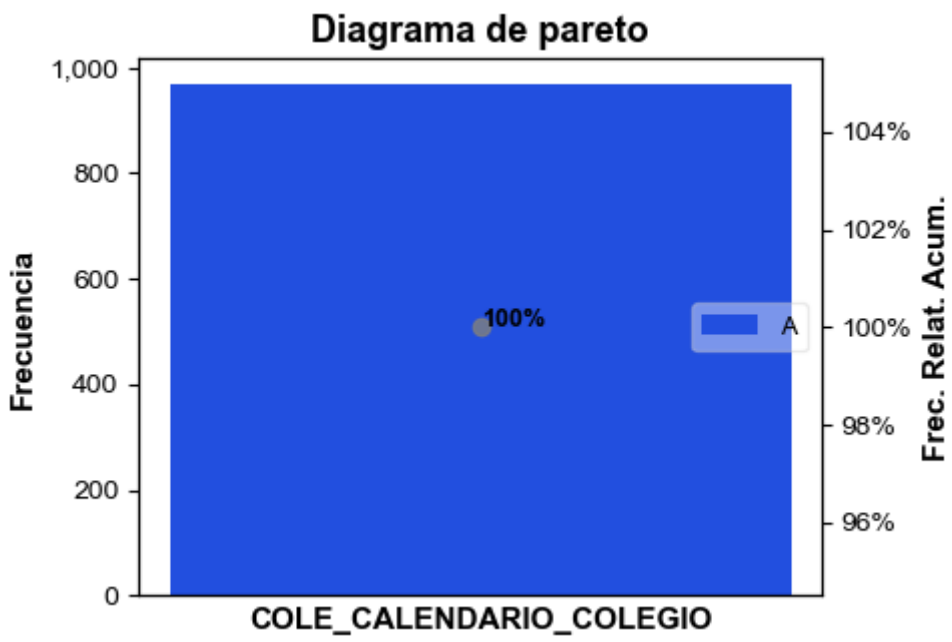


In [79]:
```
inter_uncond_descrp_cat_var(col = df['COLE_CALENDARIO_COLEGIO'])
```

▸ Estadísticos

▾ **Gráficos descriptivos**

Circular | Pareto



5.5.2.2 COLE_GENEROPOBLACION

In [80]:
```
inter_uncond_descrp_cat_var(col = df['COLE_GENEROPOBLACION'])
```

▾ **Estadísticos**

Frecuencia

Conteo de registros

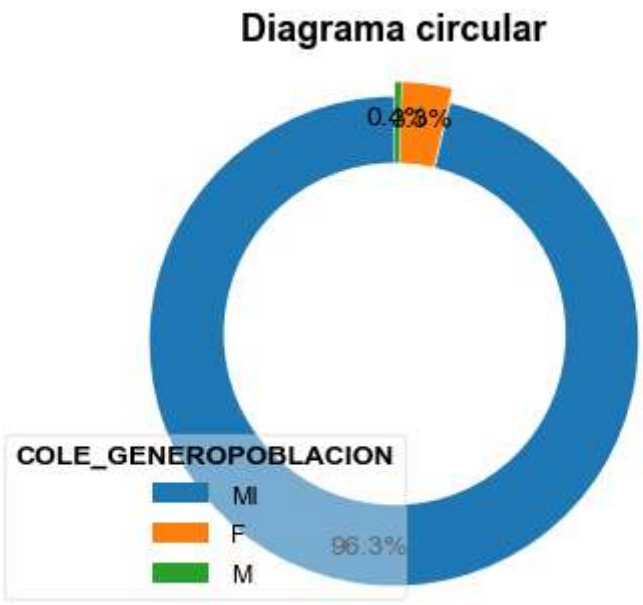
	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Con dato	968	100.00%	100.00%

Conteo de frecuencias

	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Categorías			
MI	932	96.28%	96.28%
F	32	3.31%	99.59%
M	4	0.41%	100.00%

▾ **Gráficos descriptivos**

Circular | Pareto

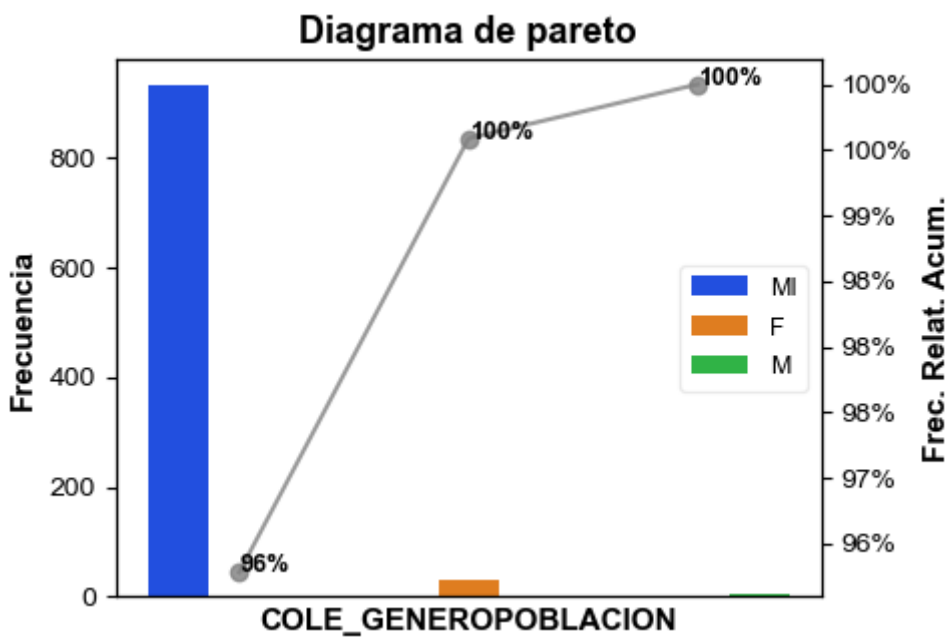


In [81]:
```
inter_uncond_descrp_cat_var(col = df['COLE_GENEROPOBLACION'])
```

▸ Estadísticos

▾ **Gráficos descriptivos**

Circular | Pareto



5.5.2.3 COLE_NATURALEZA

In [82]:
```
inter_uncond_descrp_cat_var(col = df['COLE_NATURALEZA'])
```

▾ **Estadísticos**

Frecuencia

Conteo de registros

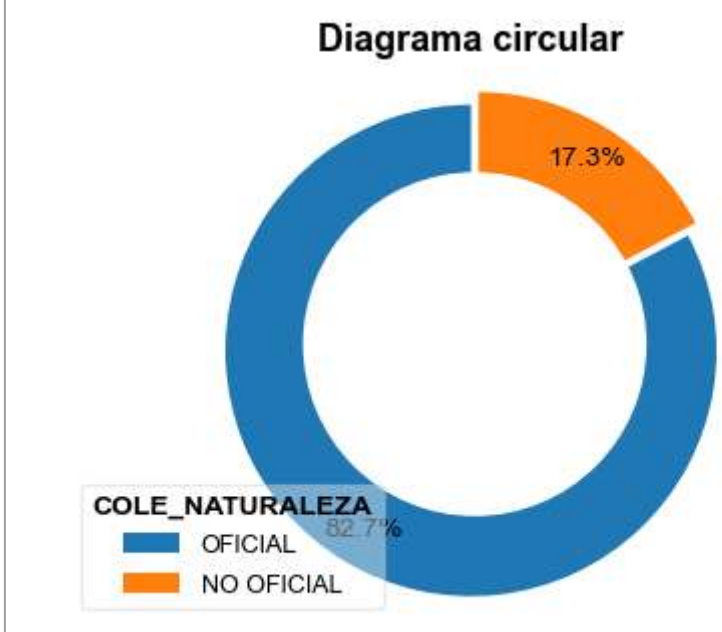
	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Con dato	968	100.00%	100.00%

Conteo de frecuencias

	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Categorías			
OFICIAL	801	82.75%	82.75%
NO OFICIAL	167	17.25%	100.00%

▾ **Gráficos descriptivos**

Circular | Pareto

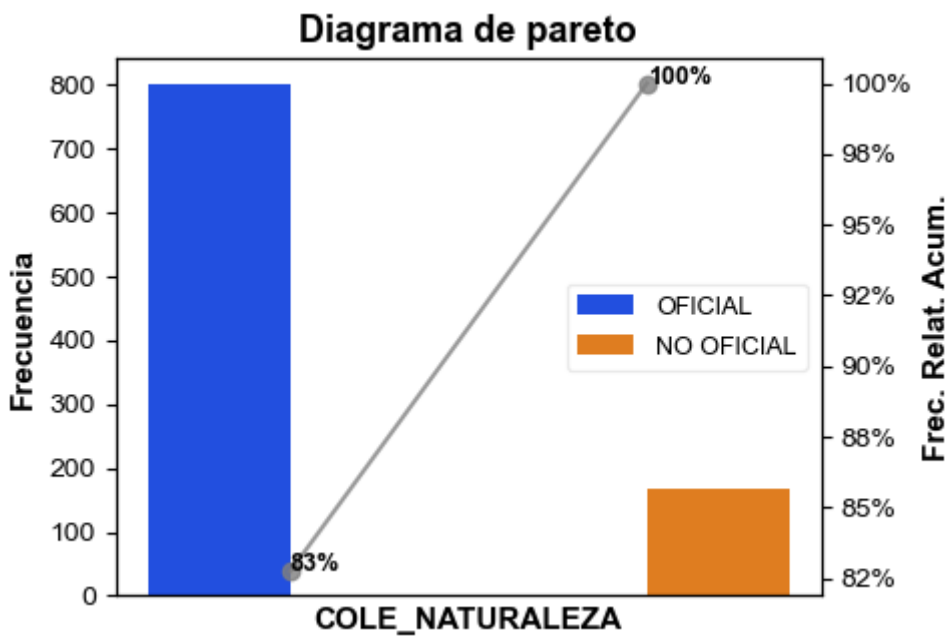


In [83]:
```
inter_uncond_descrp_cat_var(col = df['COLE_NATURALEZA'])
```

▸ Estadísticos

▾ **Gráficos descriptivos**

Circular | Pareto



5.5.2.4 COLE_CATEGORIA

In [84]:
```
inter_uncond_descrp_cat_var(col = df['COLE_CATEGORIA'])
```

▾ **Estadísticos**

Frecuencia

Conteo de registros

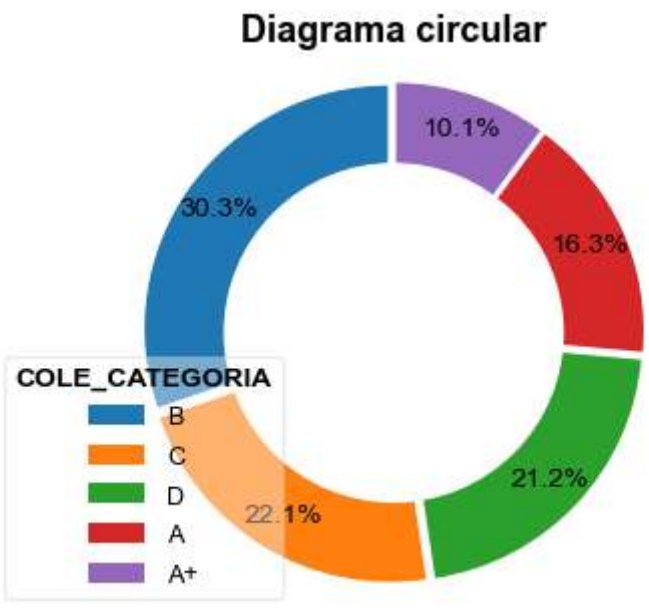
	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Con dato	968	100.00%	100.00%

Conteo de frecuencias

	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Categorías			
B	293	30.27%	30.27%
C	214	22.11%	52.38%
D	205	21.18%	73.55%
A	158	16.32%	89.88%
A+	98	10.12%	100.00%

▾ **Gráficos descriptivos**

Circular | Pareto

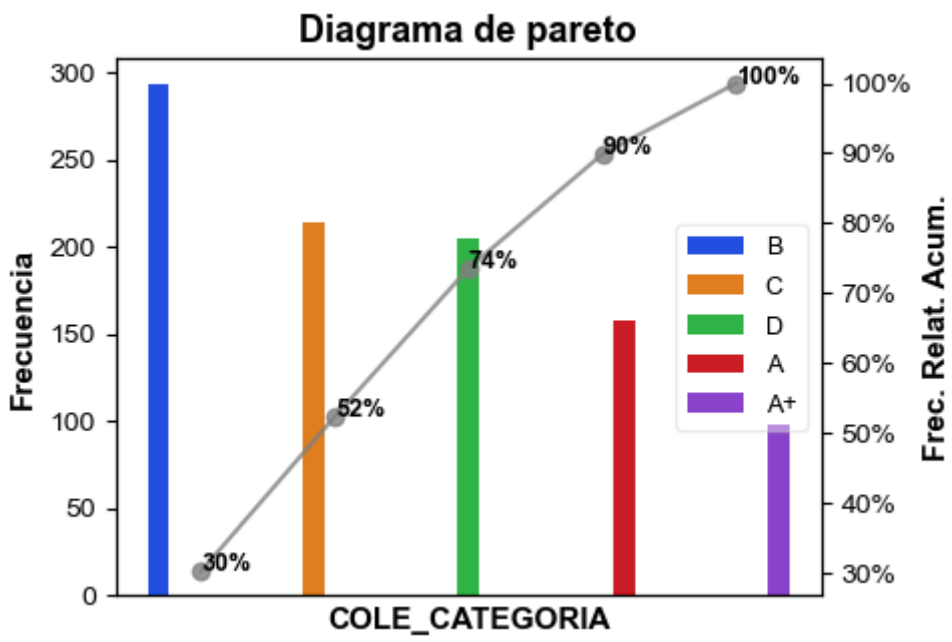


In [85]:
```
inter_uncond_descrp_cat_var(col = df['COLE_CATEGORIA'])
```

▸ Estadísticos

▾ **Gráficos descriptivos**

Circular | Pareto



5.5.2.5 ZONA

In [86]:
```
inter_uncond_descrp_cat_var(col = df['ZONA'])
```

▾ **Estadísticos**

Frecuencia

Conteo de registros

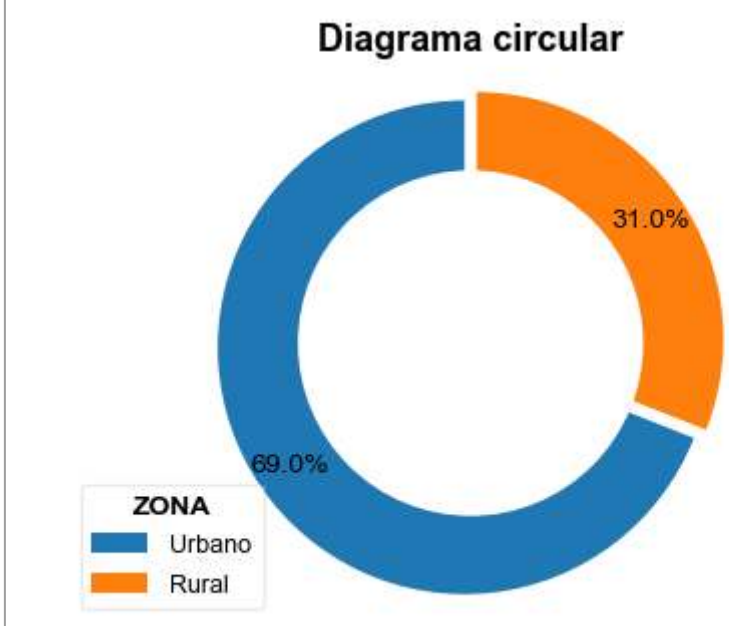
	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Con dato	968	100.00%	100.00%

Conteo de frecuencias

	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Categorías			
Urbano	668	69.01%	69.01%
Rural	300	30.99%	100.00%

▾ **Gráficos descriptivos**

Circular | Pareto

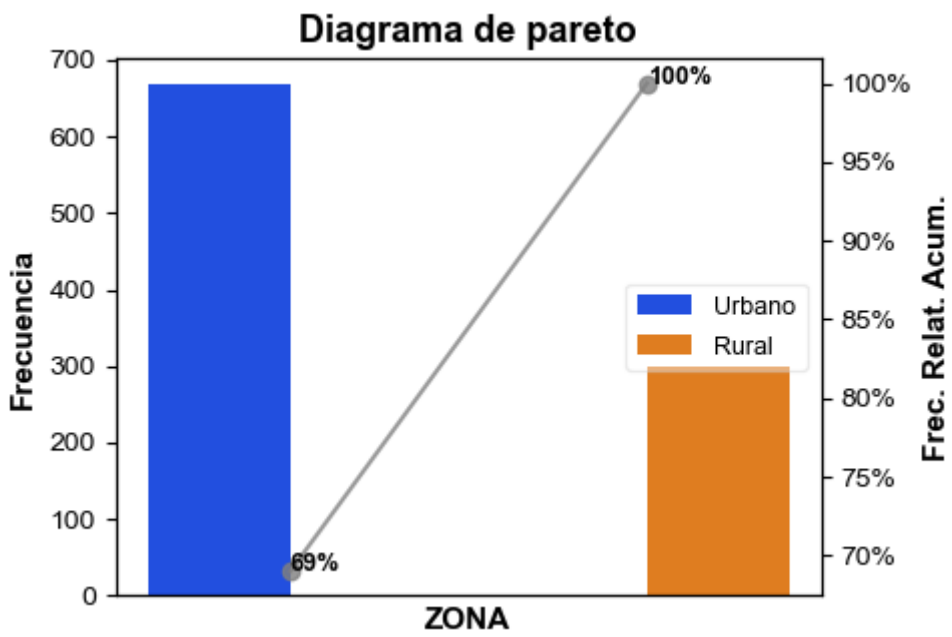


In [87]:
```
inter_uncond_descrp_cat_var(col = df['ZONA'])
```

▸ Estadísticos

▾ **Gráficos descriptivos**

Circular | Pareto



5.5.2.6 MODE_ESTRATOVIVIENDA

In [88]:
```
inter_uncond_descrp_cat_var(col = df['MODE_ESTRATOVIVIENDA'])
```

▾ **Estadísticos**

Frecuencia

Conteo de registros

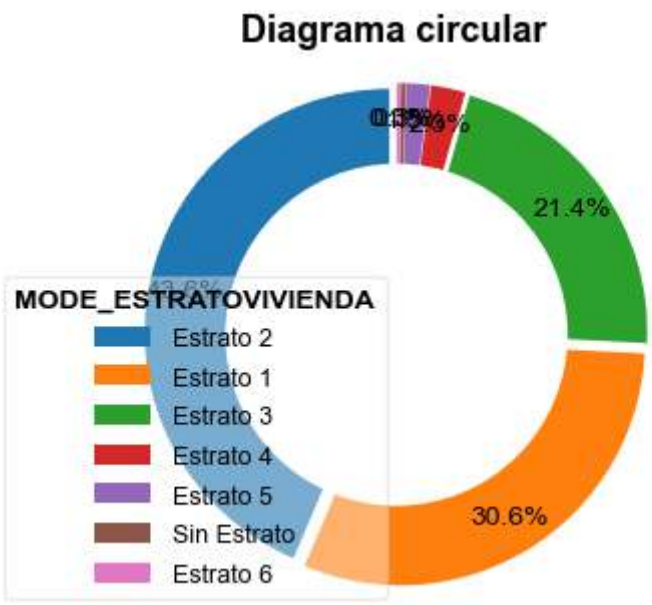
	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Con dato	968	100.00%	100.00%

Conteo de frecuencias

	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Categorías			
Estrato 2	422	43.60%	43.60%
Estrato 1	296	30.58%	74.17%
Estrato 3	207	21.38%	95.56%
Estrato 4	22	2.27%	97.83%
Estrato 5	15	1.55%	99.38%
Sin Estrato	3	0.31%	99.69%
Estrato 6	3	0.31%	100.00%

▾ **Gráficos descriptivos**

Circular | Pareto

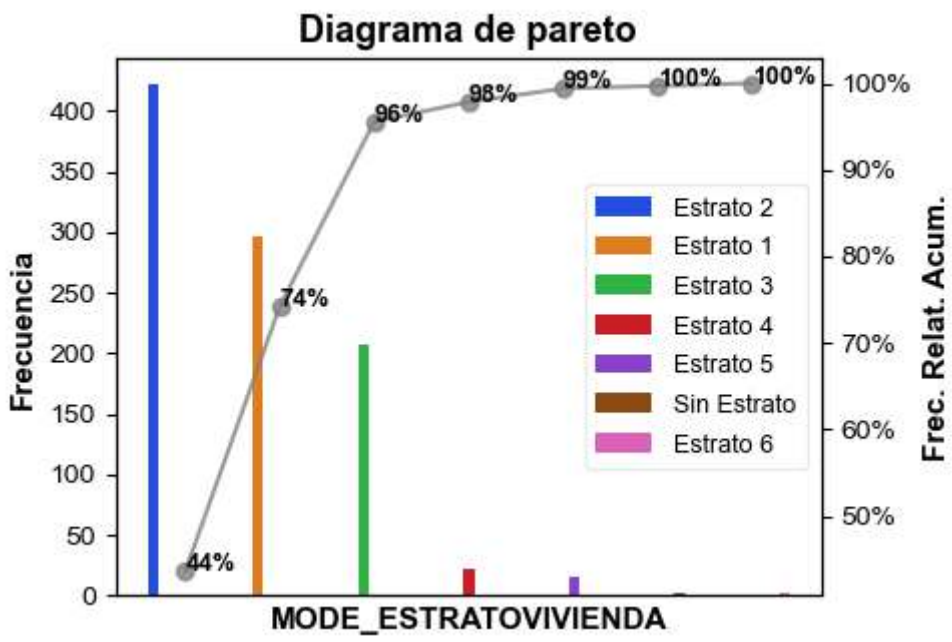


```
In [89]: inter_uncond_descrp_cat_var(col = df['MODE_ESTRATOVIVIENDA'])
```

▸ Estadísticos

▾ **Gráficos descriptivos**

Circular | Pareto



5.5.2.7 MODE_EDU_MADRE

In [90]:
```
inter_uncond_descrp_cat_var(col = df['MODE_EDU_MADRE'])
```

▾ **Estadísticos**

Frecuencia

Conteo de registros

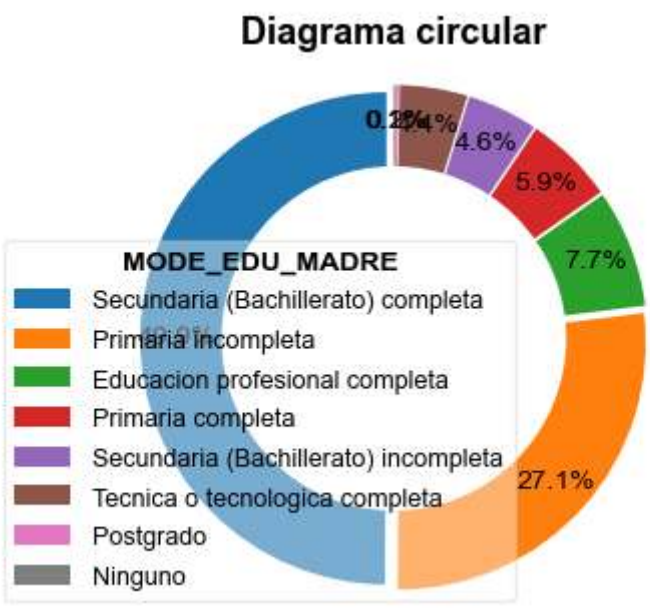
	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Con dato	968	100.00%	100.00%

Conteo de frecuencias

	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Categorías			
Secundaria (Bachillerato) completa	483	49.90%	49.90%
Primaria incompleta	262	27.07%	76.96%
Educacion profesional completa	75	7.75%	84.71%
Primaria completa	57	5.89%	90.60%
Secundaria (Bachillerato) incompleta	45	4.65%	95.25%
Tecnica o tecnologica completa	43	4.44%	99.69%
Postgrado	2	0.21%	99.90%
Ninguno	1	0.10%	100.00%

▾ **Gráficos descriptivos**

Circular | Pareto

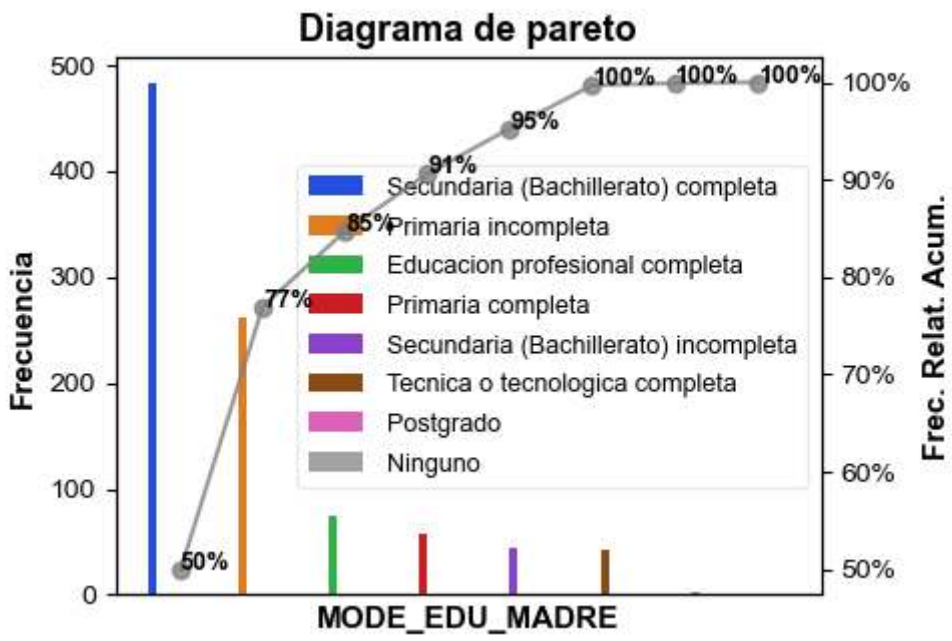


```
In [91]: inter_uncond_descrp_cat_var(col = df['MODE_EDU_MADRE'])
```

▸ Estadísticos

▾ **Gráficos descriptivos**

Circular | Pareto



5.5.2.8 MODE_EDU_PADRE

In [92]:
```
inter_uncond_descrp_cat_var(col = df['MODE_EDU_PADRE'])
```

▾ **Estadísticos**

Frecuencia

Conteo de registros

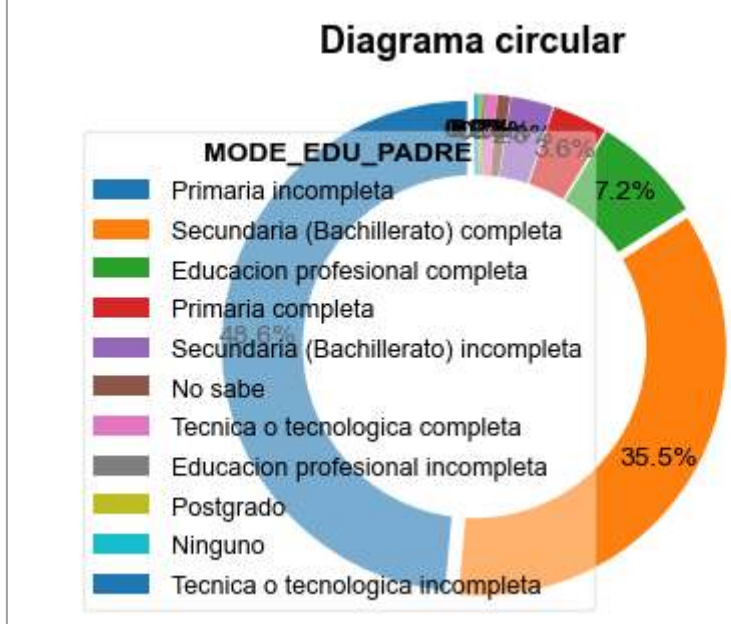
	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Con dato	968	100.00%	100.00%

Conteo de frecuencias

	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Categorías			
Primaria incompleta	470	48.55%	48.55%
Secundaria (Bachillerato) completa	344	35.54%	84.09%
Educacion profesional completa	70	7.23%	91.32%
Primaria completa	35	3.62%	94.94%
Secundaria (Bachillerato) incompleta	27	2.79%	97.73%
No sabe	8	0.83%	98.55%
Tecnica o tecnologica completa	7	0.72%	99.28%
Educacion profesional incompleta	2	0.21%	99.48%
Postgrado	2	0.21%	99.69%
Ninguno	2	0.21%	99.90%
Tecnica o tecnologica incompleta	1	0.10%	100.00%

▾ **Gráficos descriptivos**

Circular | Pareto

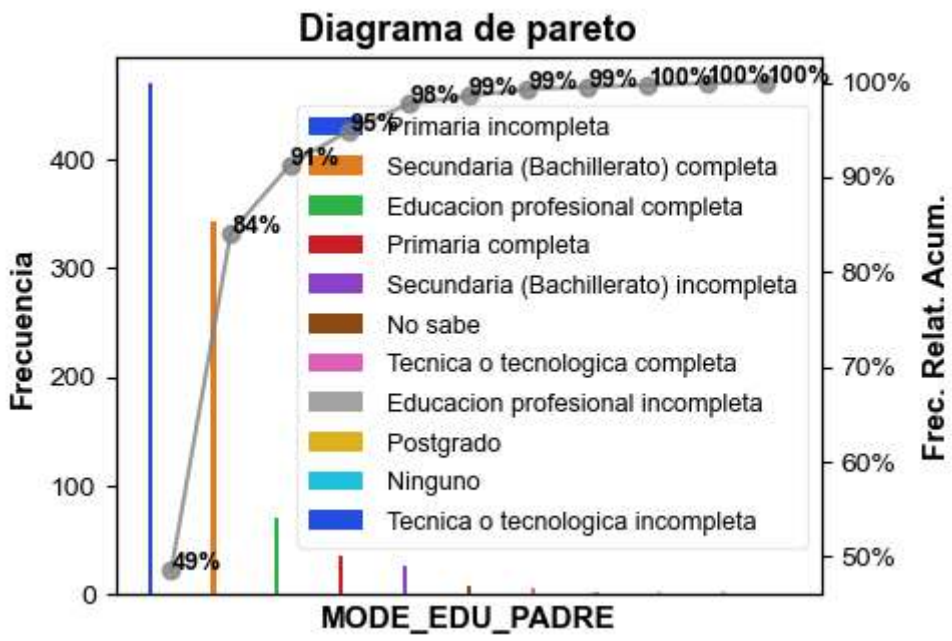


In [93]:
```
inter_uncond_descrp_cat_var(col = df['MODE_EDU_PADRE'])
```

▸ Estadísticos

▾ **Gráficos descriptivos**

Circular | Pareto



5.5.2.9 IND_BILINGUE

In [94]:
```
inter_uncond_descrp_cat_var(col = df['IND_BILINGUE'])
```

▾ **Estadísticos**

Frecuencia

Conteo de registros

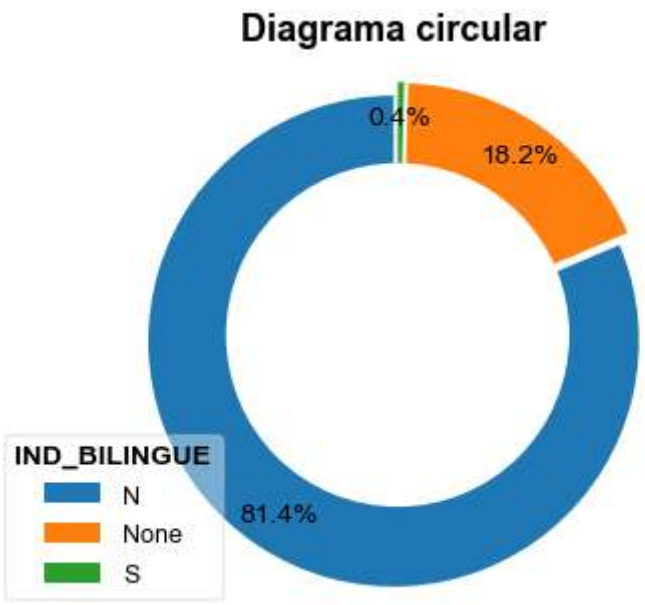
	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Con dato	968	100.00%	100.00%

Conteo de frecuencias

	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Categorías			
N	788	81.40%	81.40%
None	176	18.18%	99.59%
S	4	0.41%	100.00%

▾ **Gráficos descriptivos**

Circular | Pareto

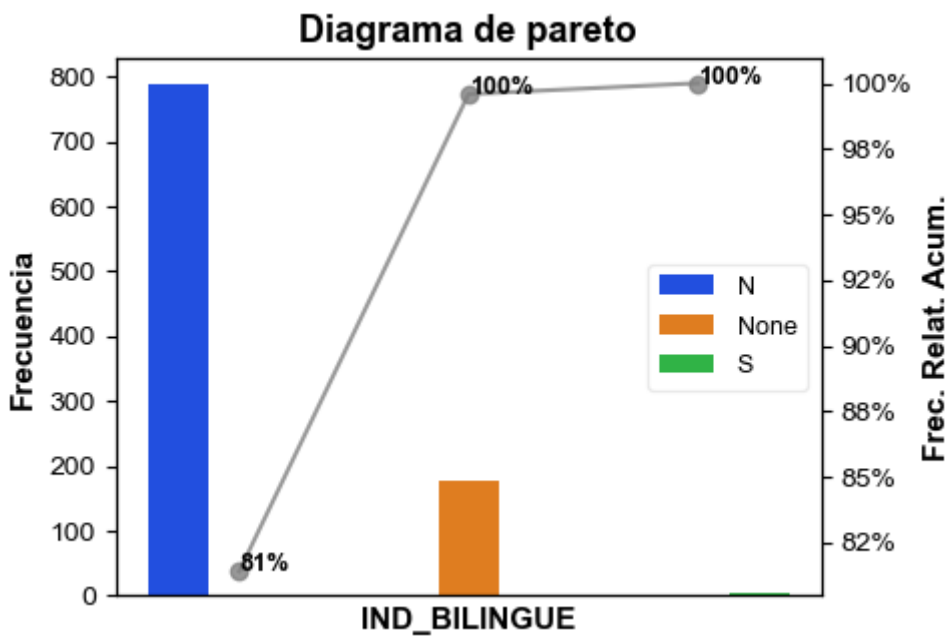


In [95]:
```
inter_uncond_descrp_cat_var(col = df['IND_BILINGUE'])
```

▸ Estadísticos

▾ **Gráficos descriptivos**

Circular | Pareto



5.5.2.10 CARACTER_IE

In [96]:
```
inter_uncond_descrp_cat_var(col = df['CARACTER_IE'])
```

▾ **Estadísticos**

Frecuencia

Conteo de registros

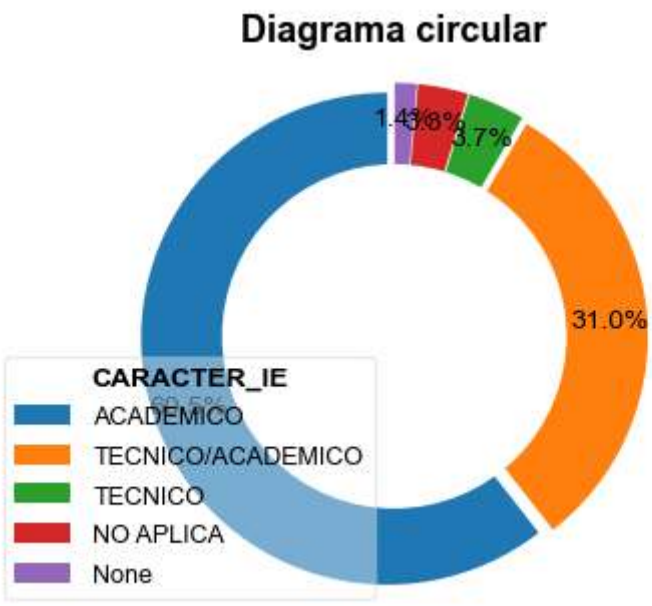
	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Con dato	968	100.00%	100.00%

Conteo de frecuencias

	Frec.Abs.	Frec.Rel.	Frec.Rel.Acum.
Categorías			
ACADEMICO	586	60.54%	60.54%
TECNICO/ACADEMICO	300	30.99%	91.53%
TECNICO	36	3.72%	95.25%
NO APLICA	32	3.31%	98.55%
None	14	1.45%	100.00%

▾ **Gráficos descriptivos**

Circular | Pareto

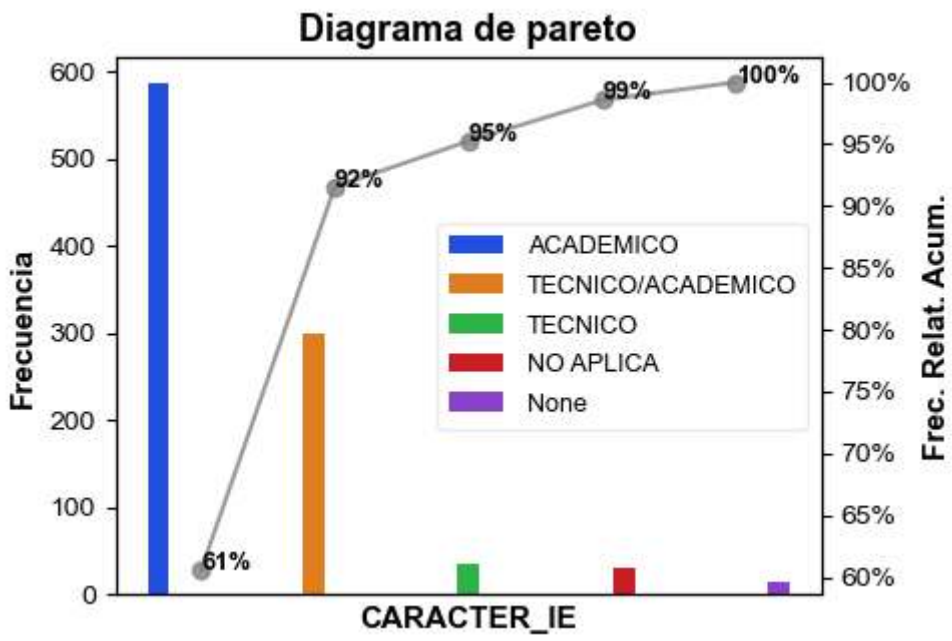


In [97]:
```
inter_uncond_descrp_cat_var(col = df['CARACTER_IE'])
```

▸ Estadísticos

▾ **Gráficos descriptivos**

Circular | Pareto



5.5.3 ◐ Detección de outliers en sistema de calificación

In [98]:
```
col_out = ['COD_COL','EVALUADOS_ULTIMOS_3', 'INDICE_MATEMATICAS',
       'INDICE_C_NATURALES', 'INDICE_SOCIALES_CIUDADANAS',
       'INDICE_LECTURA_CRITICA', 'INDICE_INGLES']
df_out = df[col_out]
df_out.set_index('COD_COL',inplace=True )
Mean_out = df_out.mean()
print(Mean_out)
df_out.head(2)
```

```
EVALUADOS_ULTIMOS_3           160.462810
INDICE_MATEMATICAS              0.666575
INDICE_C_NATURALES              0.672375
INDICE_SOCIALES_CIUDADANAS      0.666211
INDICE_LECTURA_CRITICA          0.706532
INDICE_INGLES                   0.666768
dtype: float64
```

Out[98]:

COD_COL	EVALUADOS_ULTIMOS_3	INDICE_MATEMATICAS	INDICE_C_NATURALES	INDICE_SOCIALES_CIUDADANAS	INDICE_LECTURA_CRITICA
205790000235	61	0.5400	0.5809	0.5543	0.6150
105147000568	262	0.6977	0.6650	0.6726	0.7084

In [99]:
```
nf,nc=df_out.shape
range(0,nf)
```

Out[99]: range(0, 968)

In [100]:
```
Dist_1=[]
Dist_2=[]
Dist_Fro=[]
nf,nc=df_out.shape
for i in range(0,nf):
    dist1 = np.linalg.norm(np.array(Mean_out)-np.array(df_out)[i],1)
    dist2 = np.linalg.norm(np.array(Mean_out)-np.array(df_out)[i],2)
    distFro = np.linalg.norm(np.array(Mean_out)-np.array(df_out)[i])
    Dist_1.append(dist1)
    Dist_2.append(dist2)
    Dist_Fro.append(distFro)
P90_1=np.percentile(Dist_1,90)
P90_2=np.percentile(Dist_2,90)
P90_Fro=np.percentile(Dist_Fro,90)
Dist_1=pd.DataFrame(Dist_1)
Dist_1Out=Dist_1[Dist_1[0]>P90_1]
Dist_2=pd.DataFrame(Dist_2)
Dist_2Out=Dist_2[Dist_2[0]>P90_2]
Dist_Fro=pd.DataFrame(Dist_Fro)
Dist_FroOut=Dist_Fro[Dist_Fro[0]>P90_Fro]
```

In [101]:
```
outliers(df)
```

Identificacion I.E Atipicas

N. Manhattan | N.Euclidea | N. Frobenius

TOP Mejores puntajes

	COLE_INST_NOMBRE	INDICE_TOTAL	EVALUADOS_ULTIMOS_3	INDICE_AJUST	COLE_CATEGORIA
51	INST EDUC CEFA	0.7491	2767	1.283192	A
160	INST EDUC INEM JOSE FELIX DE RESTREPO	0.7464	2282	1.254044	A
296	LIC SALAZAR Y HERRERA	0.7745	1566	1.212801	A+
111	I. E. LICEO CAUCASIA	0.7268	829	1.023305	A
225	INS TEC INDUSTRIAL PASCUAL BRAVO	0.7697	726	1.015520	A
562	I. E. ESCUELA NORMAL SUPERIOR DE MARIA	0.5551	1341	1.001840	D
248	INST EDUC CONCEJO DE MEDELLIN	0.7487	726	1.000779	A
746	INST EDUC LOLA GONZALEZ	0.7403	701	0.984419	A
670	IE LICEO ANTIOQUENO	0.7492	606	0.945332	A
756	I. E. JOSE MARIA BERNAL	0.6738	673	0.926458	B

TOP peores puntajes

	COLE_INST_NOMBRE	INDICE_TOTAL	EVALUADOS_ULTIMOS_3	INDICE_AJUST	COLE_CATEGORIA
508	I. E. R. LOS LLANOS	0.5415	9	0.034564	D
609	I. E. R. SANTA FE DE LAS PLATAS	0.4726	10	0.036119	D
954	I. E. R. VILLA FATIMA ARRIBA	0.4480	12	0.042150	D
962	I. E. R. ENRIQUE DURAN	0.5482	12	0.045904	D
499	I. E. ANTONIO ROLDAN BETANCUR	0.5314	334	0.628410	D
209	I.E. TURBO	0.5777	315	0.634127	D
521	I. E. DE JESUS	0.5462	339	0.640641	D
908	I. E. SAN FERNANDO	0.6452	316	0.669287	C
718	I. E. R. MARIA DEL ROSARIO	0.5805	353	0.671285	D
155	INST EDUC VILLA DEL SOCORRO	0.6711	310	0.675936	B

In [102]:
```
outliers(df)
```

Identificacion I.E Atipicas

N. Manhattan | N.Euclidea | N. Frobenius

TOP Mejores puntajes

	COLE_INST_NOMBRE	INDICE_TOTAL	EVALUADOS_ULTIMOS_3	INDICE_AJUST	COLE_CATEGORIA
51	INST EDUC CEFA	0.7491	2767	1.283192	A
160	INST EDUC INEM JOSE FELIX DE RESTREPO	0.7464	2282	1.254044	A
296	LIC SALAZAR Y HERRERA	0.7745	1566	1.212801	A+
111	I. E. LICEO CAUCASIA	0.7268	829	1.023305	A
225	INS TEC INDUSTRIAL PASCUAL BRAVO	0.7697	726	1.015520	A
562	I. E. ESCUELA NORMAL SUPERIOR DE MARIA	0.5551	1341	1.001840	D
248	INST EDUC CONCEJO DE MEDELLIN	0.7487	726	1.000779	A
746	INST EDUC LOLA GONZALEZ	0.7403	701	0.984419	A
670	IE LICEO ANTIOQUENO	0.7492	606	0.945332	A
756	I. E. JOSE MARIA BERNAL	0.6738	673	0.926458	B

TOP peores puntajes

	COLE_INST_NOMBRE	INDICE_TOTAL	EVALUADOS_ULTIMOS_3	INDICE_AJUST	COLE_CATEGORIA
508	I. E. R. LOS LLANOS	0.5415	9	0.034564	D
609	I. E. R. SANTA FE DE LAS PLATAS	0.4726	10	0.036119	D
499	I. E. ANTONIO ROLDAN BETANCUR	0.5314	334	0.628410	D
209	I.E. TURBO	0.5777	315	0.634127	D
521	I. E. DE JESUS	0.5462	339	0.640641	D
908	I. E. SAN FERNANDO	0.6452	316	0.669287	C
718	I. E. R. MARIA DEL ROSARIO	0.5805	353	0.671285	D
155	INST EDUC VILLA DEL SOCORRO	0.6711	310	0.675936	B
704	I. E. PUEBLO NUEVO	0.5532	386	0.683973	D
366	I. E. SAN LUIS GONZAGA	0.6871	313	0.687243	B

In [103]:
```
outliers(df)
```

Identificacion I.E Atipicas

N. Manhattan | N.Euclidea | N. Frobenius

TOP Mejores puntajes

	COLE_INST_NOMBRE	INDICE_TOTAL	EVALUADOS_ULTIMOS_3	INDICE_AJUST	COLE_CATEGORIA
51	INST EDUC CEFA	0.7491	2767	1.283192	A
160	INST EDUC INEM JOSE FELIX DE RESTREPO	0.7464	2282	1.254044	A
296	LIC SALAZAR Y HERRERA	0.7745	1566	1.212801	A+
111	I. E. LICEO CAUCASIA	0.7268	829	1.023305	A
225	INS TEC INDUSTRIAL PASCUAL BRAVO	0.7697	726	1.015520	A
562	I. E. ESCUELA NORMAL SUPERIOR DE MARIA	0.5551	1341	1.001840	D
248	INST EDUC CONCEJO DE MEDELLIN	0.7487	726	1.000779	A
746	INST EDUC LOLA GONZALEZ	0.7403	701	0.984419	A
670	IE LICEO ANTIOQUENO	0.7492	606	0.945332	A
756	I. E. JOSE MARIA BERNAL	0.6738	673	0.926458	B

TOP peores puntajes

	COLE_INST_NOMBRE	INDICE_TOTAL	EVALUADOS_ULTIMOS_3	INDICE_AJUST	COLE_CATEGORIA
508	I. E. R. LOS LLANOS	0.5415	9	0.034564	D
609	I. E. R. SANTA FE DE LAS PLATAS	0.4726	10	0.036119	D
499	I. E. ANTONIO ROLDAN BETANCUR	0.5314	334	0.628410	D
209	I.E. TURBO	0.5777	315	0.634127	D
521	I. E. DE JESUS	0.5462	339	0.640641	D
908	I. E. SAN FERNANDO	0.6452	316	0.669287	C
718	I. E. R. MARIA DEL ROSARIO	0.5805	353	0.671285	D
155	INST EDUC VILLA DEL SOCORRO	0.6711	310	0.675936	B
704	I. E. PUEBLO NUEVO	0.5532	386	0.683973	D
366	I. E. SAN LUIS GONZAGA	0.6871	313	0.687243	B

6. ◉ Etapa 5: Modelamiento del sistema de recomendación

6.1. ◎ Selección de Parámetros

Se seleccionan los parámetros sobre los cuales se realizaran los modelos.

se crea una copia del DataFrame para trabajar en los modelos

In [104]:
```
df_modelos = df.copy()
```

visualizamos los parametros disponibles

In [105]:
```
df_modelos.columns
```

Out[105]:
```
Index(['COD_COL', 'COLE_INST_NOMBRE', 'COLE_DEPTO_COLEGIO',
       'COLE_CALENDARIO_COLEGIO', 'COLE_GENEROPOBLACION',
       'EVALUADOS_ULTIMOS_3', 'COLE_NATURALEZA', 'INDICE_MATEMATICAS',
       'INDICE_C_NATURALES', 'INDICE_SOCIALES_CIUDADANAS',
       'INDICE_LECTURA_CRITICA', 'INDICE_INGLES', 'INDICE_TOTAL',
       'COLE_CATEGORIA', 'NOMBRE_DEPARTAMENTO', 'NOMBRE_MUNICIPIO', 'ZONA',
       'LATITUD', 'LONGITUD', 'MODE_ESTRATOVIVIENDA', 'IND_EST_TRAB',
       'MODE_EDU_MADRE', 'MODE_EDU_PADRE', 'IND_BILINGUE', 'CARACTER_IE',
       'INDICE_AJUST'],
      dtype='object')
```

seleccionamos las variables

In [106]:
```
#variables = ['COD_COL','COLE_CALENDARIO_COLEGIO','COLE_GENEROPOBLACION','COLE_CATEGORIA','ZONA']
#variables = ['COD_COL','COLE_GENEROPOBLACION','COLE_NATURALEZA','ZONA','MODE_ESTRATOVIVIENDA','CARACTER_IE']
variables = ['COD_COL','COLE_GENEROPOBLACION','MODE_ESTRATOVIVIENDA','CARACTER_IE']
df_modelos = df[variables]
```

se convierten variables categoricas a Dummies

In [107]:
```
df_modelos = pd.get_dummies(df_modelos, columns = variables[1:])
df_modelos.head(3)
```

Out[107]:

	COD_COL	COLE_GENEROPOBLACION_F	COLE_GENEROPOBLACION_M	COLE_GENEROPOBLACION_MI	MODE_ESTRATOVIVIENDA_Estrato 1	M
0	205790000235	0	0	1	1	
1	105147000568	0	0	1	0	
2	105147000401	0	0	1	0	

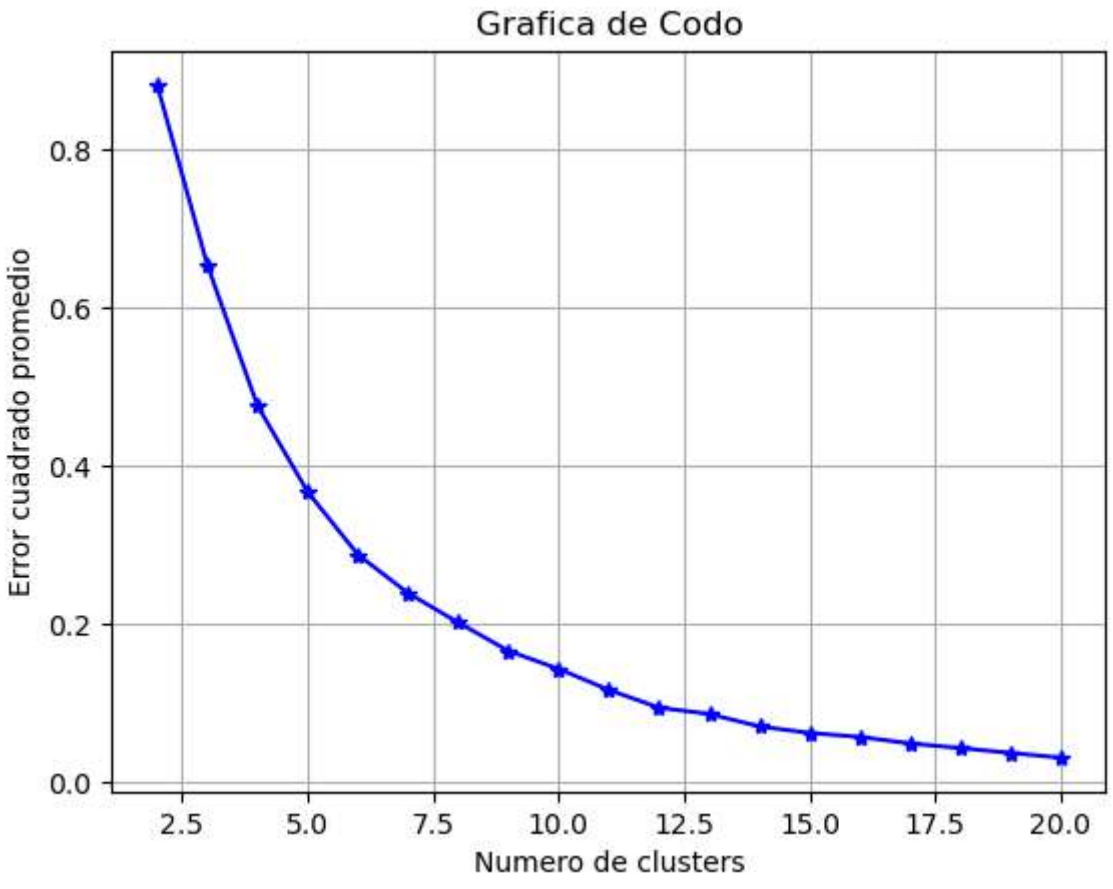
6.2. ◎ k-means

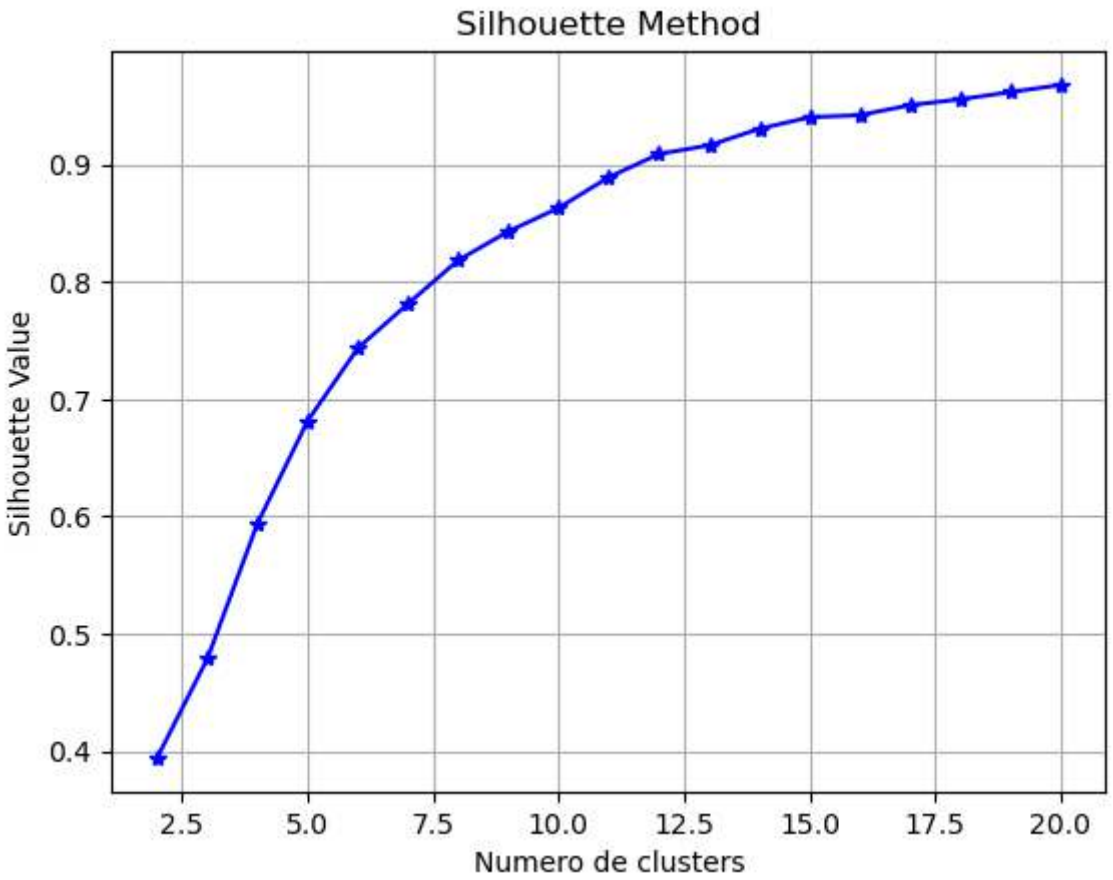
6.2.1 ◐ Modelo - Métodos de selección de k

Se corre el modelo de K-means para diferentes valores de **K** (número de clusters) y evaluando que tan adecuada es cada valor de k midiendo la distancia de os puntos a los centroides de su respectivo cluster, entre mejor segmentados estén los grupos menor será esta distancia.

Si se divide en demasiados clusters el modelo perdería sentido y si se divide en pocos clusters las distancias serían muy grandes, es por esto que aplicamos los siguientes métodos conocidos como el método del codo y el método de Silhouette.

In [108]:
```
k_algs, k_res = k_selection(df_modelos.drop(['COD_COL'], axis = 1), 2, 20)
```





6.2.2 ◐ Selección de k

In [109]:
```
k = 8
```

In [110]:
```
algorithm = k_algs[k-2] #
clustering = k_res[k-2] #
```

In [111]:
```
cls_list = algorithm.predict(df_modelos.drop(['COD_COL'], axis = 1))
```

In [112]:
```
df_kmeans = df_modelos.copy()
```

In [113]:
```
df_kmeans.insert(len(df_modelos.columns), 'Cluster', cls_list, True)
df_kmeans.head(3)
```

Out[113]:

	COD_COL	COLE_GENEROPOBLACION_F	COLE_GENEROPOBLACION_M	COLE_GENEROPOBLACION_MI	MODE_ESTRATOVIVIENDA_Estrato 1	M
0	205790000235	0	0	1	1	
1	105147000568	0	0	1	0	
2	105147000401	0	0	1	0	

6.2.3 ◐ Evaluación de Resultados

1. COLE_GENEROPOBLACION
2. MODE_ESTRATOVIVIENDA
3. CARACTER_IE

In [114]:
```
df_eval = df_kmeans.copy()
df_eval['COLE_GENEROPOBLACION'] = df.loc[:,'COLE_GENEROPOBLACION']
df_eval['MODE_ESTRATOVIVIENDA'] = df.loc[:,'MODE_ESTRATOVIVIENDA']
df_eval['CARACTER_IE'] = df.loc[:,'CARACTER_IE']
df_eval.head(3)
```

Out[114]:

	COD_COL	COLE_GENEROPOBLACION_F	COLE_GENEROPOBLACION_M	COLE_GENEROPOBLACION_MI	MODE_ESTRATOVIVIENDA_Estrato 1	M
0	205790000235	0	0	1	1	
1	105147000568	0	0	1	0	
2	105147000401	0	0	1	0	

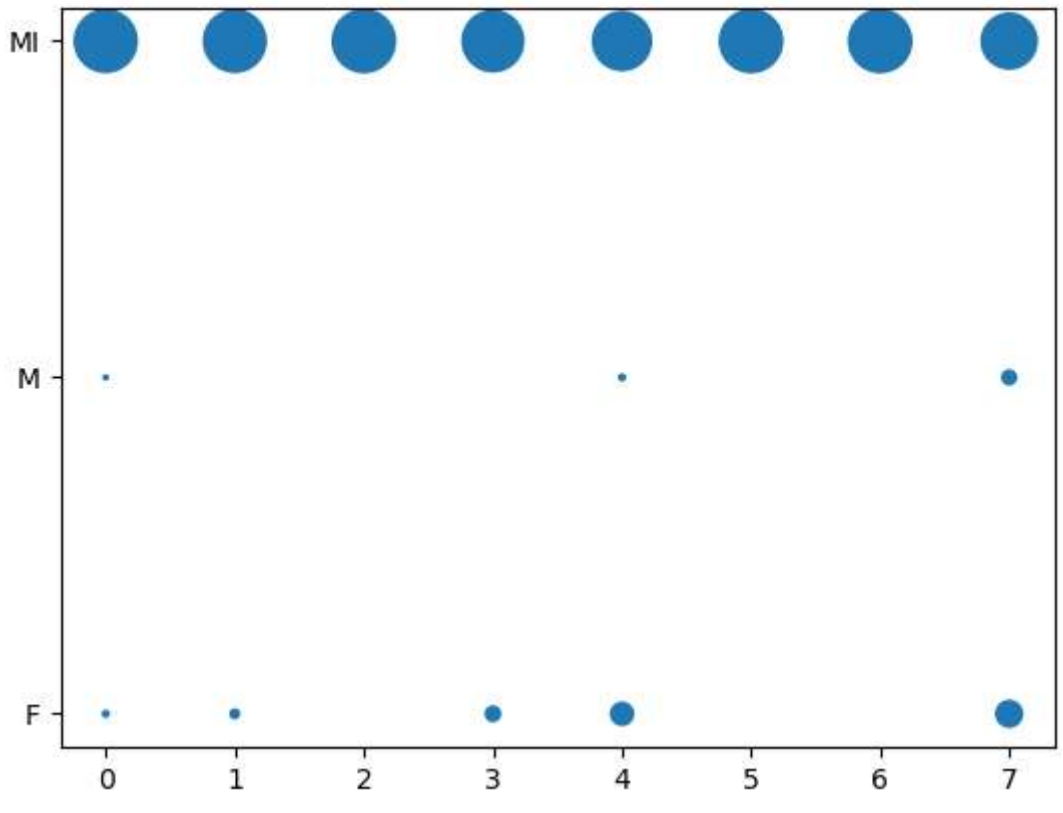
6.2.3.1 ◤ COLE_GENEROPOBLACION

In [115]:
```
tbl_pvt,tbl_frm = tabla_cluster(df_eval,'COLE_GENEROPOBLACION')
tbl_pvt
```

Out[115]:

Cluster	0	1	2	3	4	5	6	7
COLE_GENEROPOBLACION								
F	2	3	0	4	16	0	0	7
M	1	0	0	0	1	0	0	2
MI	253	165	198	74	112	74	24	32

In [116]:
```
cluster_scatter_plot(tbl_frm,'COLE_GENEROPOBLACION')
```



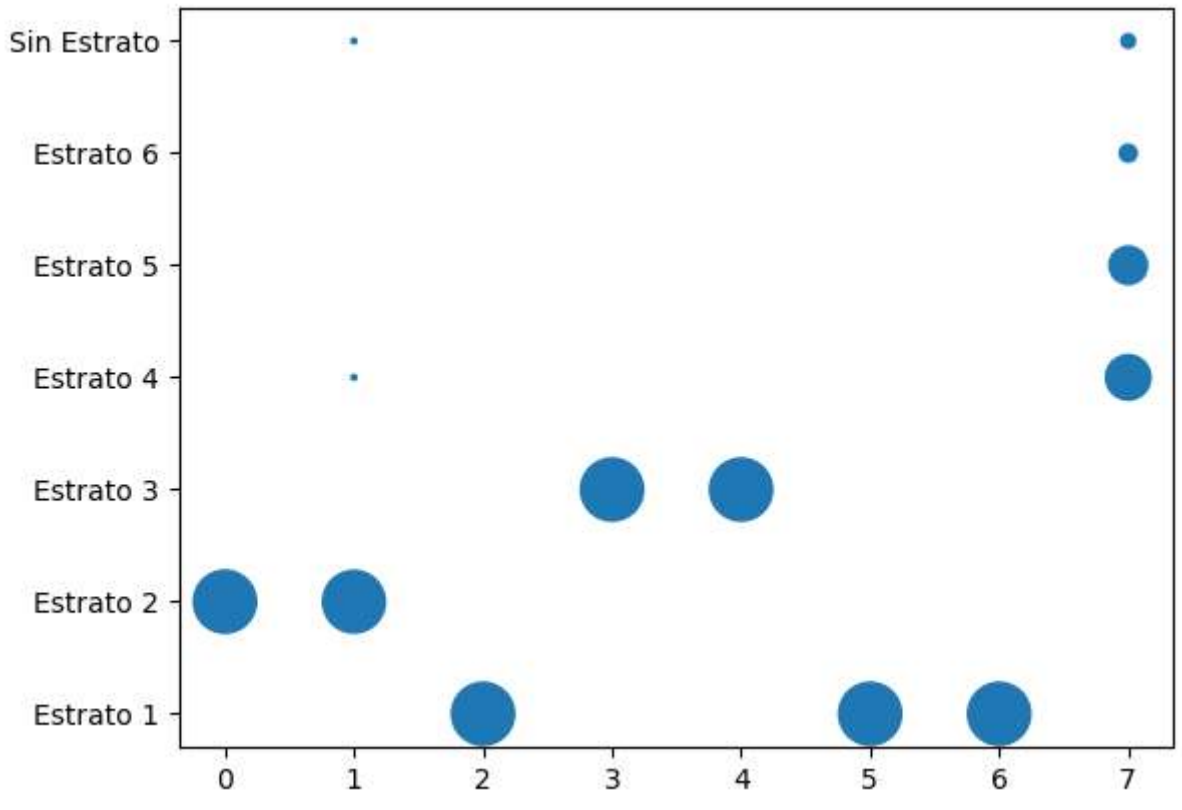
6.2.3.2 MODE_ESTRATOVIVIENDA

In [117]:
```
tbl_pvt,tbl_frm = tabla_cluster(df_eval,'MODE_ESTRATOVIVIENDA')
tbl_pvt
```

Out[117]:

Cluster	0	1	2	3	4	5	6	7
MODE_ESTRATOVIVIENDA								
Estrato 1	0	0	198	0	0	74	24	0
Estrato 2	256	166	0	0	0	0	0	0
Estrato 3	0	0	0	78	129	0	0	0
Estrato 4	0	1	0	0	0	0	0	21
Estrato 5	0	0	0	0	0	0	0	15
Estrato 6	0	0	0	0	0	0	0	3
Sin Estrato	0	1	0	0	0	0	0	2

In [118]:
```
cluster_scatter_plot(tbl_frm,'MODE_ESTRATOVIVIENDA')
```



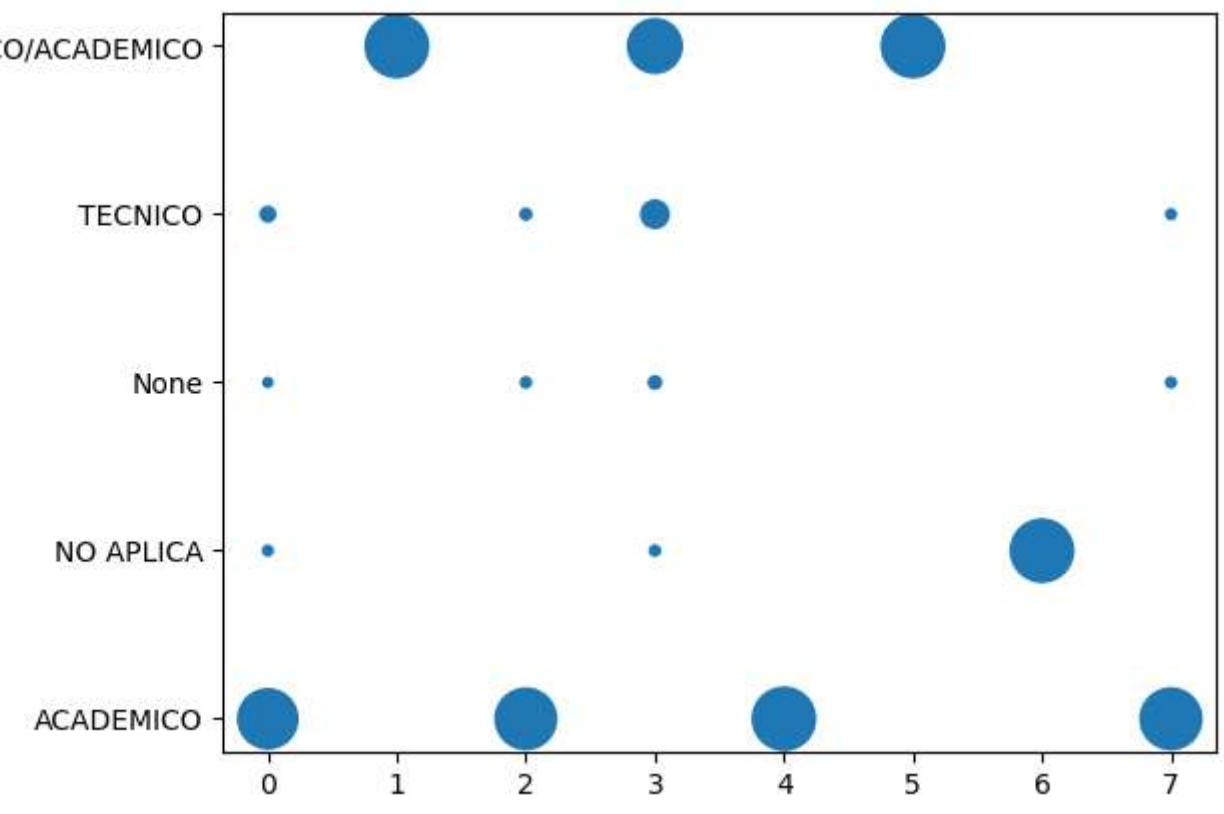
6.2.3.3 CARACTER_IE

In [119]:
```
tbl_pvt,tbl_frm = tabla_cluster(df_eval,'CARACTER_IE')
tbl_pvt
```

Out[119]:

Cluster	0	1	2	3	4	5	6	7
CARACTER_IE								
ACADEMICO	231	0	187	0	129	0	0	39
NO APLICA	6	0	0	2	0	0	24	0
None	5	0	5	3	0	0	0	1
TECNICO	14	0	6	15	0	0	0	1
TECNICO/ACADEMICO	0	168	0	58	0	74	0	0

In [120]:
```
cluster_scatter_plot(tbl_frm,'CARACTER_IE')
```



6.2.4 Prueba Clasificacion

In [121]:
```
#se extrae un registro para prueba
prueba = np.array(df_kmeans.drop(['COD_COL','Cluster'], axis = 1).iloc[0,:]).reshape(1, -1)
print(prueba)
```

```
[[0 0 1 1 0 0 0 0 0 0 1 0 0 0 0]]
```

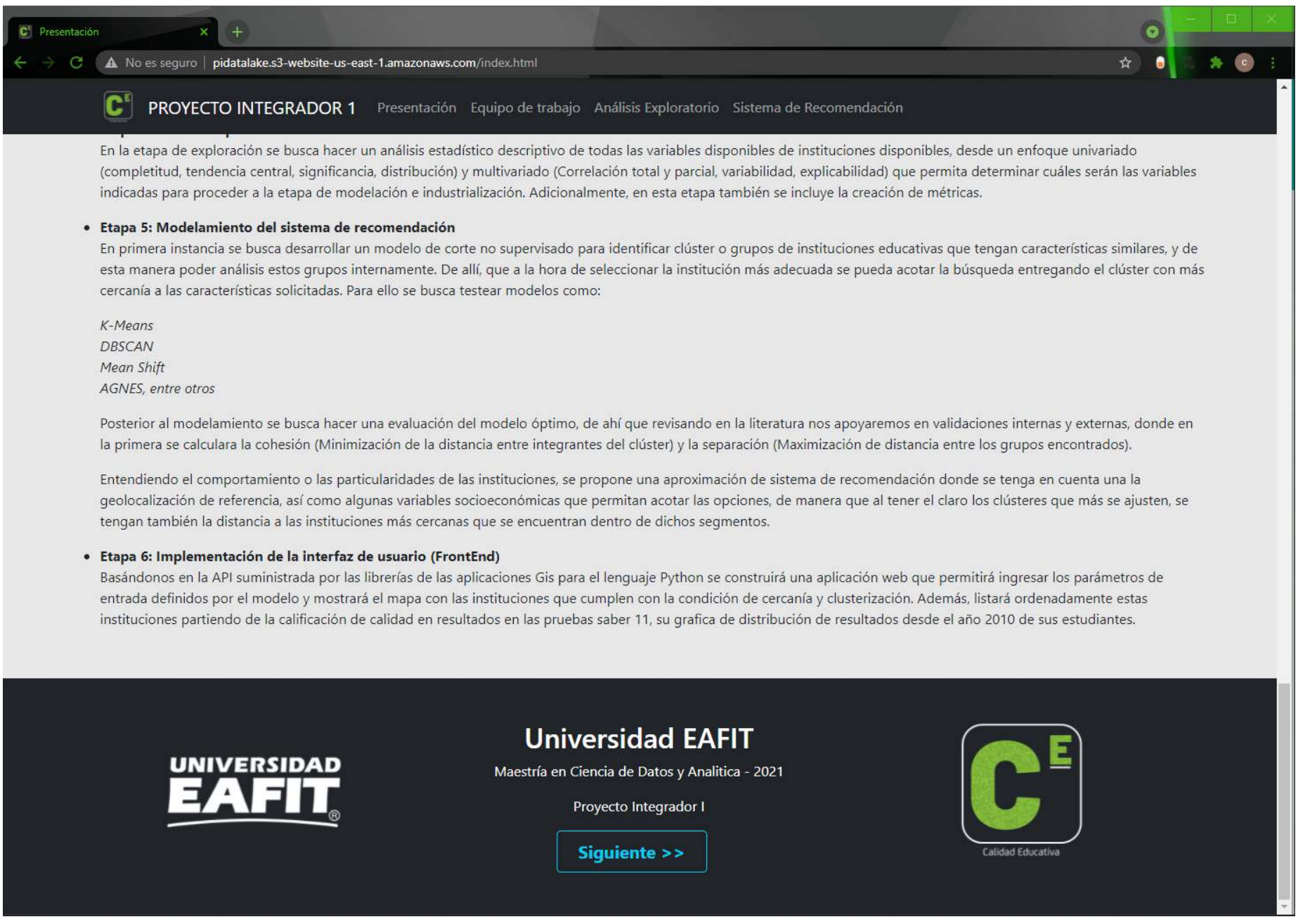
In [122]:
```
#se predice el cluster al que pertenece
algorithm.predict(prueba)[0]
```

Out[122]: 2

---

7. ◉ Etapa 6: Implementación de la interfaz de usuario (FrontEnd)

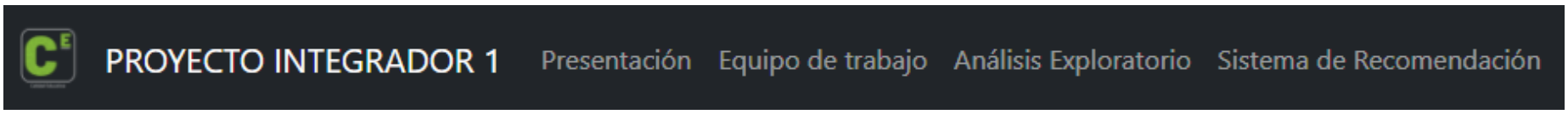
Finalmente entramos en la etapa que busca poner al alcance de un usuario los desarrollos hasta ahora implementados, para esto se ha puesto a disposición un portal web que permite al usuario buscar instituciones educativas partiendo de una posición y unas características (parámetros) deseadas. Este sistema de recomendación se basa en el modelo de clusterisacion no supervisada implementado previamente en este notebook.



7.1. ◎ Barra de Navegacion

---

a travez de la barra de navegacion podra acceder a las diferentes paginas del portal web.

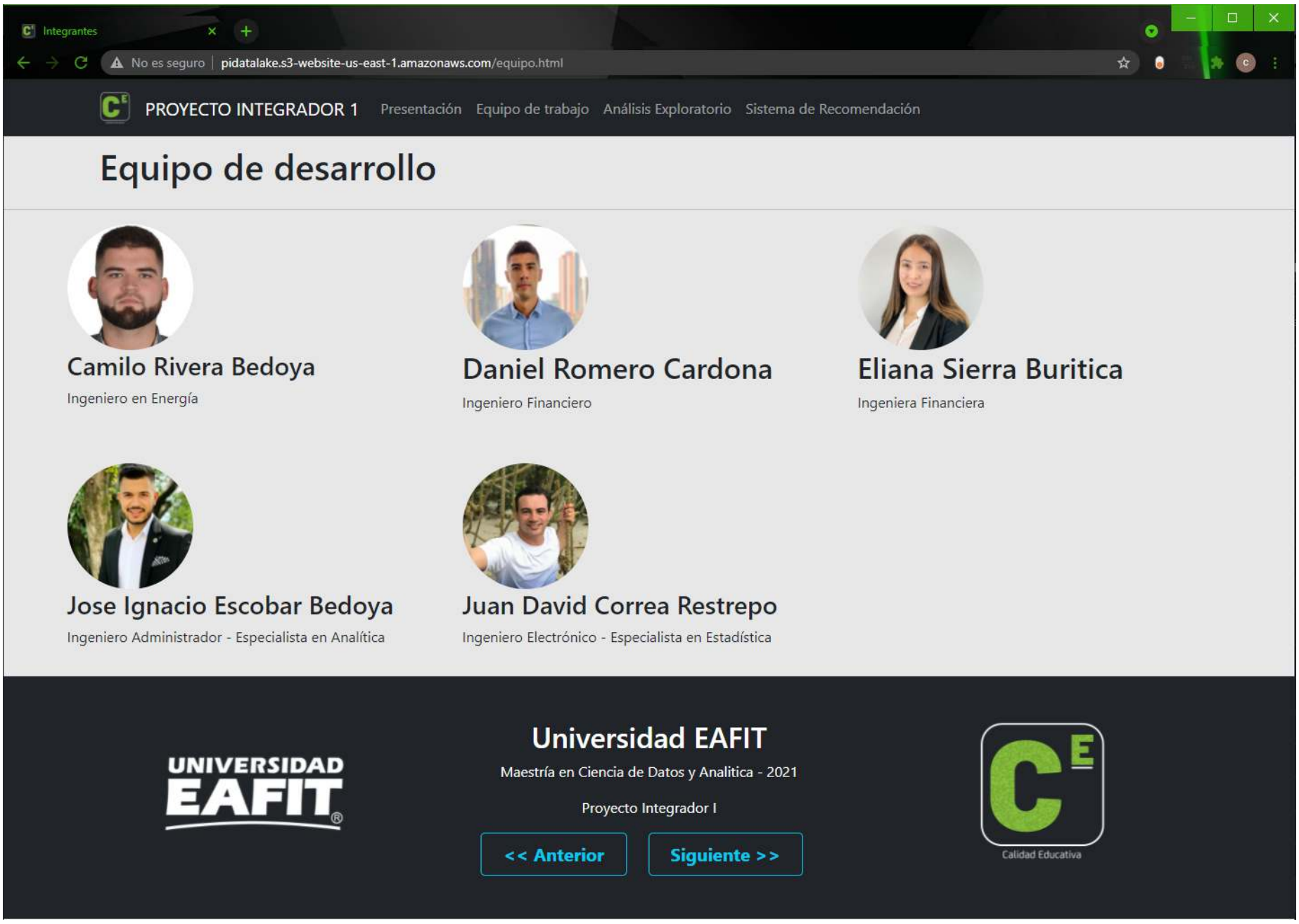


7.2. ◎ Presentación

en esta pagina podra encontrar todas las generalidades del proyecto

7.3. ◎ Equipo de Trabajo

aca podra encontrar el equipo de desarrollo del proyecto integrador



7.4. ◎ Análisis Exploratorio

---

en esta página podra encontrar el presente documento en el cual se explica paso a paso la totalidad del desarrollo del proyecto.

7.5. ◎ Sistema de Recomendación

1. Recomendadas por Cercanía
2. Recomendadas por Caracteristicas
3. Mapa
4. Ranking de planteles
5. Calificación de los planteles en cada linea de profundización

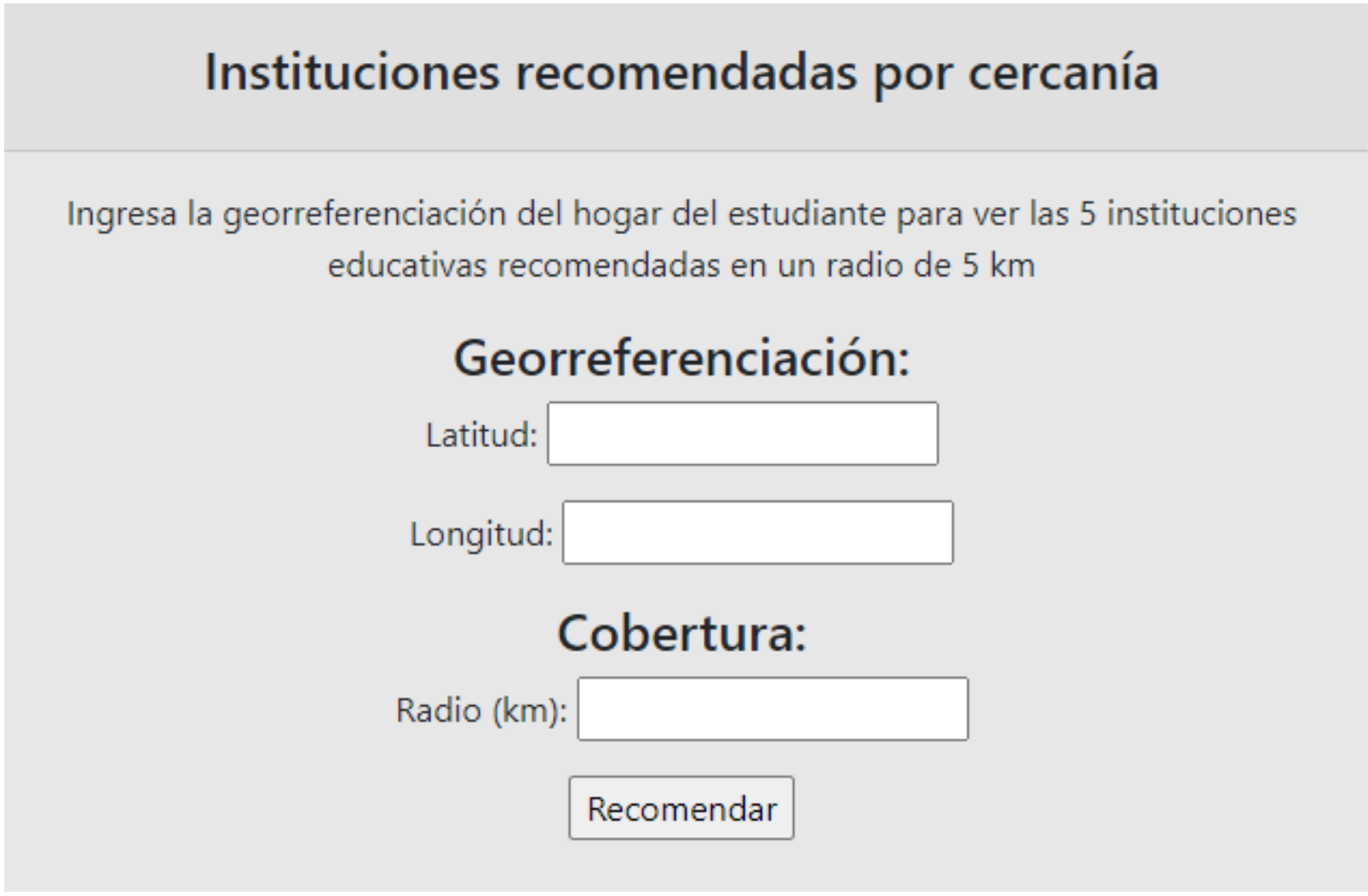
---

en esta pagina se podra hacer uso del sistema de recomendacion desarrollado.

7.5.1 ◐ Recomendadas por Cercanía

---

por este metodo se filtraran los colegior cercanos a una posicion con un rango dado por el usuario



7.5.2 ◐ Recomendadas por Caracteristicas

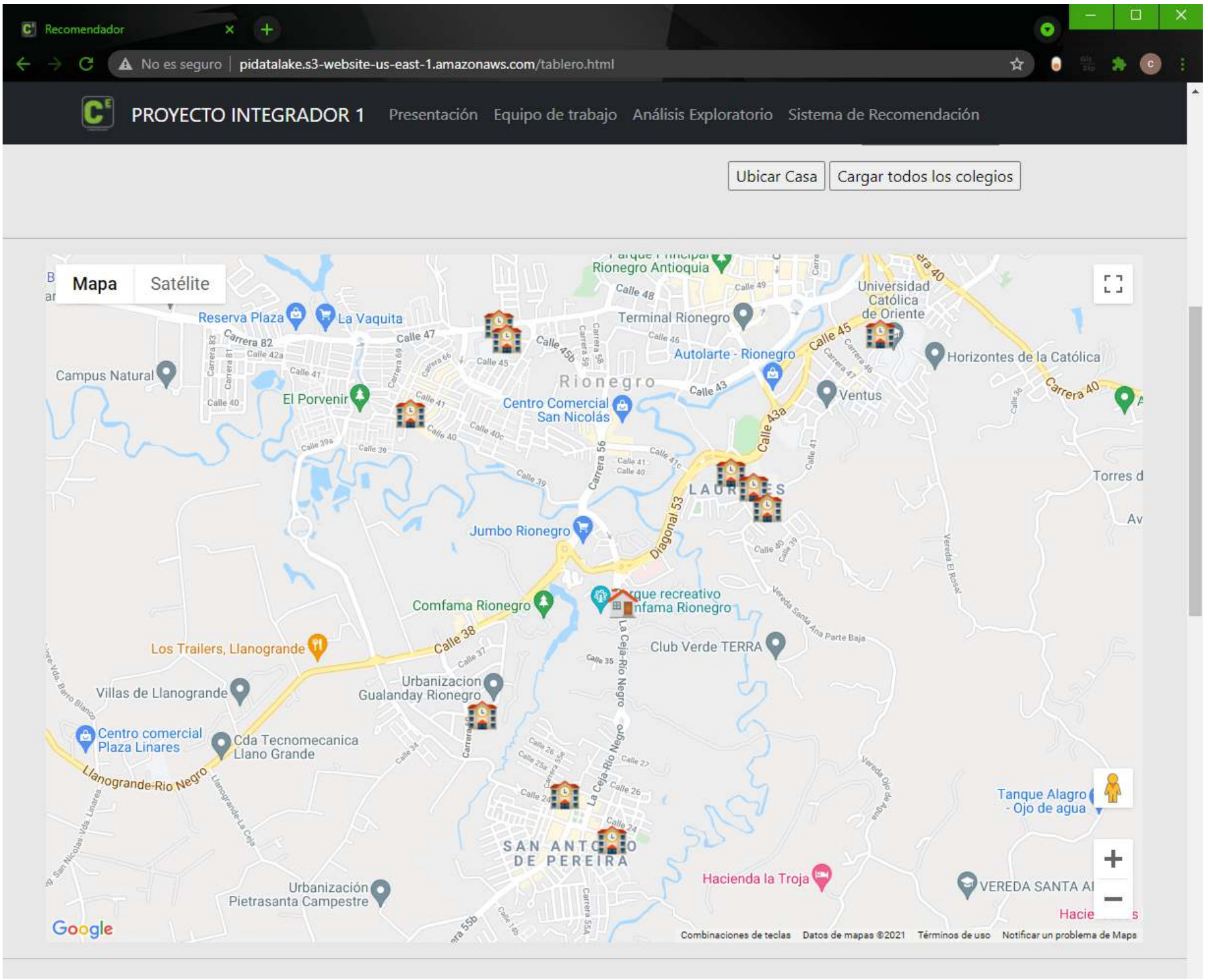
---

por este metodo se filtraran los colegior cercanos a una posicion con un rango dado por el usuario, teniendo en cuenta las caracteristicas del usuario y a que cluster obtenido por el metodo de clasificacion no supervizada ajusta mas el perfil.

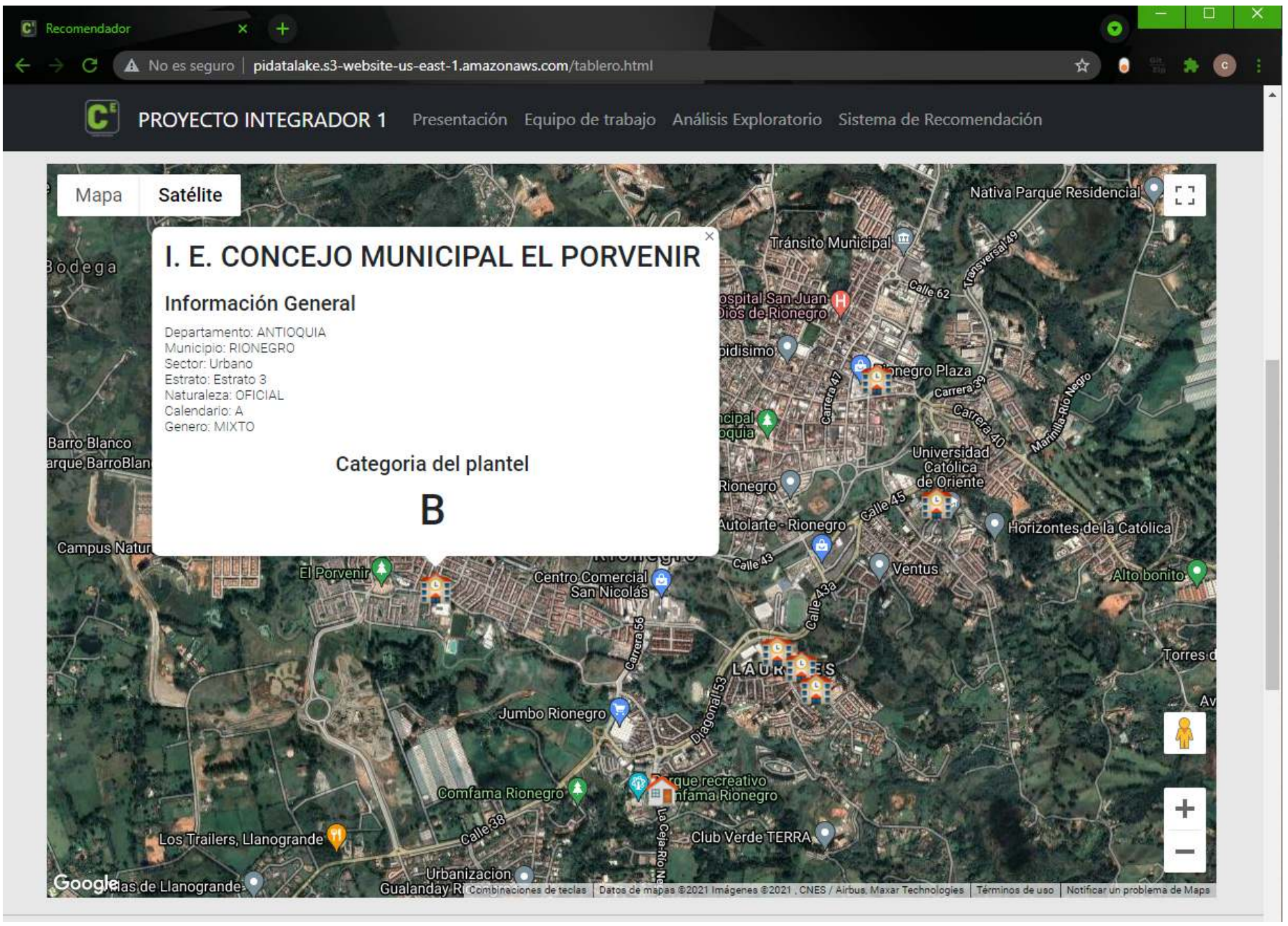
7.5.3 ◐ Mapa

---

una vez el usuario realiza una busqueda en el mapa podra visualizar su casa y las instituciones educativas que cumplen con las caracteristicas de su busqueda en el rrango especificado.



si hacemos clic sobre cualquier institucion educativa nos mostrara la informacion de la misma



7.5.4 ◐ Ranking de planteles

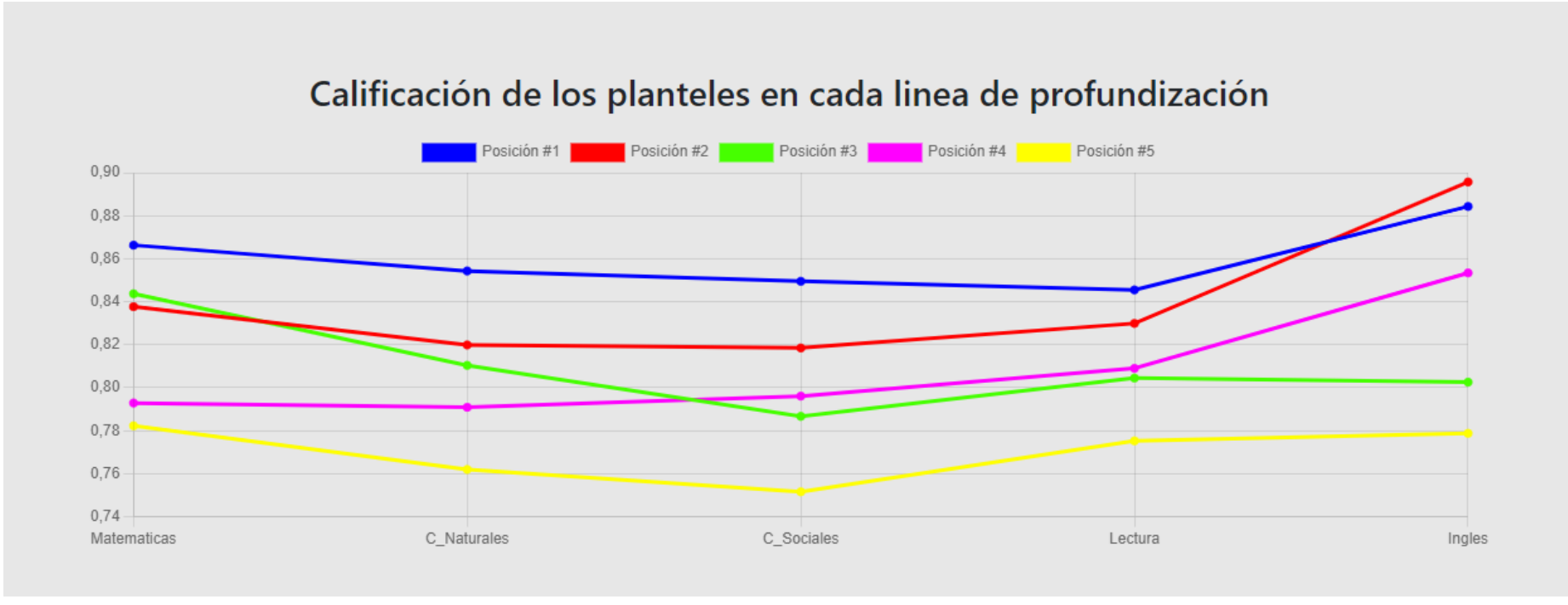
debajo del mapa se encontraran listadas las 5 instituciones educativas de mayor ranking dentro de los reasultados mostrados en el mapa

Ranking de planteles educativos con mayor calidad

Posición	Nombre	Naturaleza	Genero	Estrato	Municipio	Sector	Cat	Total
1	COLEGIO MONSE�OR ALFONSO URIBE JARAMILLO	NO OFICIAL	MIXTO	Estrato 3	RIONEGRO	Urbano	A+	0.8563
2	COLEGIO EL TRIANGULO	NO OFICIAL	MIXTO	Estrato 4	RIONEGRO	Urbano	A+	0.8318
3	I. E. TECNICO INDUSTRIAL SANTIAGO DE ARMA	OFICIAL	MIXTO	Estrato 3	RIONEGRO	Urbano	A+	0.8107
4	COLEGIO LA PRESENTACION	NO OFICIAL	MIXTO	Estrato 3	RIONEGRO	Urbano	A+	0.8015
5	COLEGIO SAN ANTONIO DE PEREIRA	NO OFICIAL	MIXTO	Estrato 3	RIONEGRO	Urbano	A	0.7686

7.5.5 ◐ Calificación de los planteles en cada linea de profundización

finalmente encontrara una grafica en la que se comparan el top 5 de instituciones en cada una de las lineas de profundizacion evaluadas en el ICFES

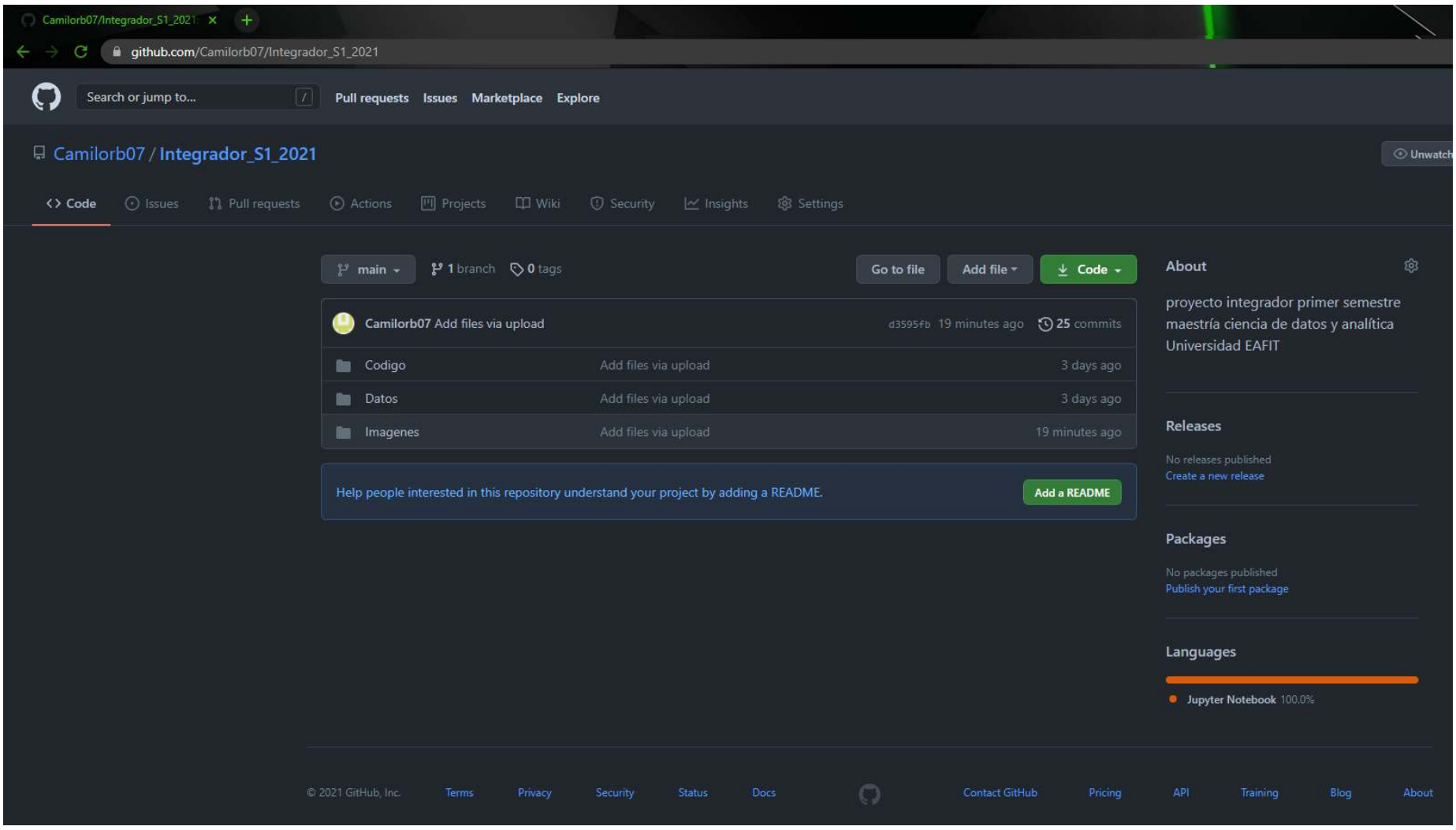


8. ◉ Repositorio

todo lo desarrollado en este trabajo se encuentra almacenado en un repositorio publico que podra encontrar en el siguiente enlace:

Repositorio (https://github.com/Camilorb07/Integrador_S1_2021)

https://github.com/Camilorb07/Integrador_S1_2021 (https://github.com/Camilorb07/Integrador_S1_2021)


Camilorb07/Integrador_S1_2021
github.com/Camilorb07/Integrador_S1_2021
Search or jump to...
Pull requests
Issues
Marketplace
Explore
Camilorb07 / Integrador_S1_2021
Unwatch
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
Camilorb07 Add files via upload
d3595fb 19 minutes ago
25 commits
Codigo
Add files via upload
3 days ago
Datos
Add files via upload
3 days ago
Imagenes
Add files via upload
19 minutes ago
Help people interested in this repository understand your project by adding a README.
Add a README
About
proyecto integrador primer semestre maestría ciencia de datos y analítica Universidad EAFIT
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Languages
Jupyter Notebook 100.0%
© 2021 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About