SELLO DE
EMPRESA
RSA
RESPONSABILIDAD SOCIAL DE ARAGÓN
2025
CEPYME PYMES Y AUTÓNOMOS DE ARAGÓN
CCOO comisiones obreras de Aragón
CEOE ARAGÓN
UGT Aragón
GOBIERNO DE ARAGON

Proyecto Jose Luis Padilla Villanova- Marzo 2025

Proyect EDA - Exploratory Data Analysis

ANALISIS MEDIOAMBIENTAL RSA2025

Introducción

Problema de negocio: Detectar empresas interesadas en nuestra asesoría medioambiental en las 1573 entidades que han conseguido el sello RSA2025 basándonos en su formulario.

Plateamiento problema técnico: Clusterizar las entidades y proponer el cluster objetivo como cliente potencial de nuestro servicios.

Crear un modelo para preseleccionar potenciales clientes.

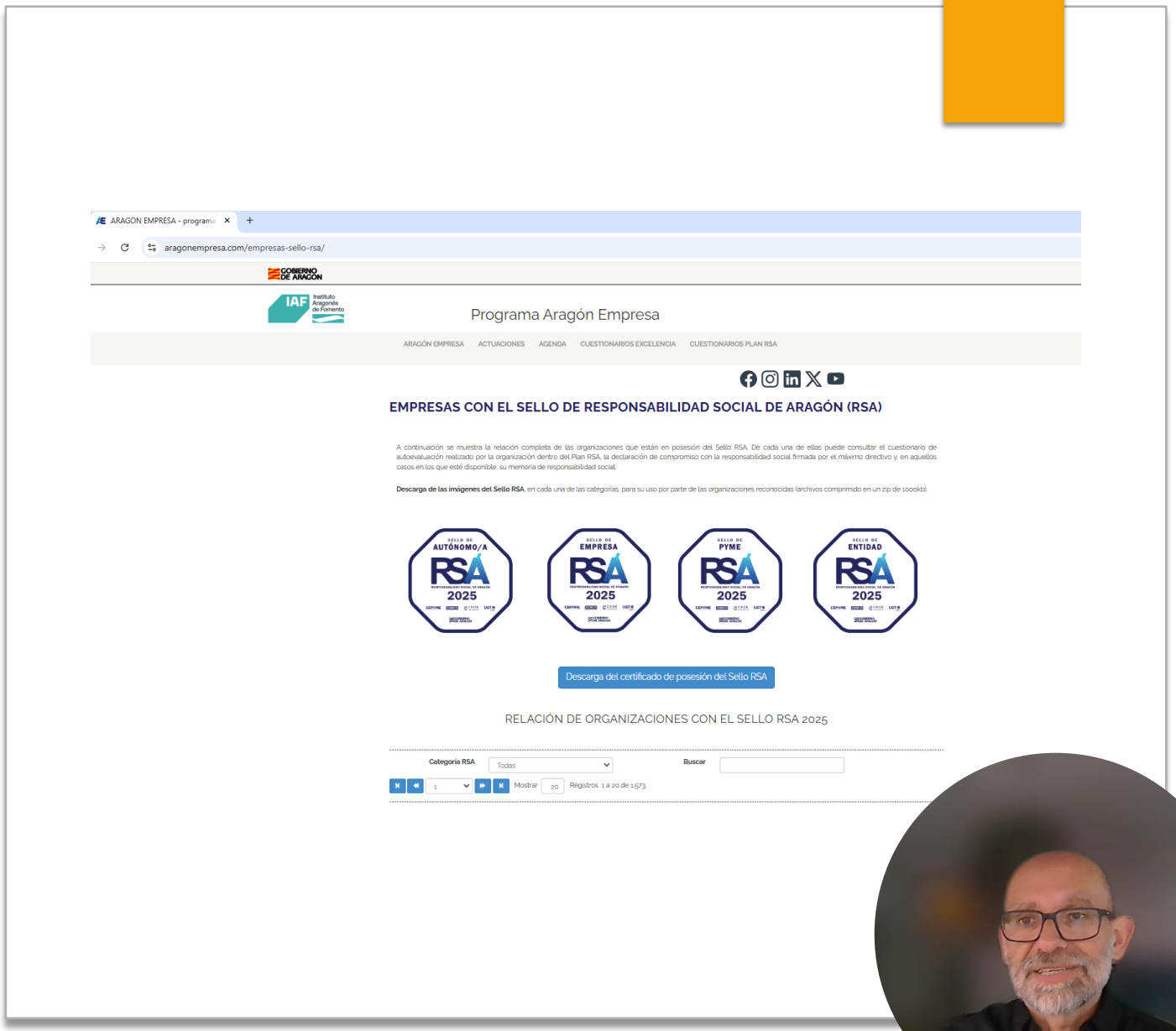
Proyecto Jose Luis Padilla Villanova- Marzo 2025

Hipótesis de partida

- Clusterización de entidades:
  - Campaña 2025: La campaña de marketing se orienta a la venta de nuestros servicios de asesoría medioambiental a las 1573 entidades que has obtenido sello RSA y de las que disponemos de un formulario con datos relevantes tanto externos como internos de la entidad.
  - Negocio demanda una lista con las entidades más predispuestas a contratar nuestro servicios para realizar un contacto directo.
- Modelo pre-selección
  - Partiendo de la clasificación realizada extrapolamos los datos con un modelo que partiendo de datos externos nos den cierta fiabilidad para determinar si encaja nuestro producto en otras entidades distintas a estas 1573, con el objetivo de rentabilidad los recursos.

Origen de los datos

- IAF https://www.aragonempresa.com/empresas-sello-rsa/



Extracción datos: Conexión

3. Extraction of initial data. Web scraping (BeautiSoup) and requests

```
# We analyze the web page from which we want to extract the data:
url = "https://www.aragonempresa.com/empresas-sello-rsa/"
response = requests.get(url)
soup = BeautifulSoup(response.text, "lxml")
```

✓ 0.8s

```
# We confirm successful connection:
if response.status_code == 200:
    print("Connection ok!")
else:
    print(f"Error: {response.status_code}")
```

✓ 0.0s

```
Connection ok!
```

Extracción datos: Inspección web

```python
# Display the HTML content of the web page in a more structured and readable format.
# This is especially useful for inspecting the page structure and locating the tags and classes needed to extract specific data.
# We see the rendered HTML with clear indentation, making it easier to identify elements such as <html>, <head>, or any other tag on the web page.
# Since the content is too long to view in the terminal, we redirect it to a text file for more convenient analysis:

# Set the path and file name to the desired folder.
ruta = "../utils/pagina_pretty.html"

with open(ruta, "w", encoding="utf-8") as file:
    file.write(soup.prettify())

print("File saved in:", ruta)
```

✓ 0.0s Python

```
File saved in: ../utils/pagina_pretty.html
```

```python
# We detect where the data we want to extract is stored:
# <a href="imprimir.php?idusuario=987&amp;idencuesta=4" class="btn btn-default" target="_blank"><img src="../img/iconosanexos/icon_pdf.png" style="height:22px;">  
# We identify that the PDF document for each entity is composed of the following string:
# imprimir.php?idusuario=987&amp;idencuesta=4
# where the number 987 is the company ID and the number 4 is the questionnaire ID, since there are 4 types of questionnaires.
# So we can extract the data for each company with the following code:
for seccion in soup.find_all("a", href=True):
    if "imprimir.php" in seccion['href']:
        print(seccion['href'])
```

✓ 0.0s Python

```
imprimir.php?idusuario=3125&idencuesta=4
imprimir.php?idusuario=1727&idencuesta=4
```

Extracción datos: Inspección web

```python
# The page has a page selector, so we can iterate through all the pages to extract all the data:
# It has a fixed structure of 20 elements per page, so we can iterate 20 at a time to extract all the data.
# We create an empty list to store the company ID and questionnaire data so we can download the PDFs.

lista_id = []

for i in range(0, 1580, 20):
    url = f"https://www.aragonempresa.com/empresas-sello-rsa/?start={i}&count=20"
    response = requests.get(url)
    soup = BeautifulSoup(response.text, "lxml")
    for seccion in soup.find_all("a", href=True):
        if "imprimir.php" in seccion['href']:
            match_usuario = re.search(r'idusuario=(\d+)&', seccion['href'])
            match_encuesta = re.search(r'idencuesta=(\d+)', seccion['href'])
            if match_usuario and match_encuesta:
                lista_id.append((match_usuario.group(1), match_encuesta.group(1)))
```

✓ 1m 0.0s

```python
# We have a list with client_id and form_id, two variables required to call
# the specific web address of each company form to download the PDF form.
# We sort by client_id
lista_id.sort()
# Example of the first element of our list
lista_id[0]
```

✓ 0.0s

```
('1000', '7')
```

Extracción datos: Descarga formularios pdf, extracción datos y verificación

3.2 PDF Form Download, Data Extraction, and Verification Process (Run Locally Only)

It is recommended not to run this section on GitHub, as it generates large files. I've provided the code and data extraction verifications for the four types of forms. We generate the initial data frame (df_0), which we save and can load

```python
# Directory where we save the PDF forms, it is a folder that we do not upload to the repository due to its size
carpeta_destino = r"..\data_ignore\EDA"

# we make sure the destination folder exists
os.makedirs(carpeta_destino, exist_ok=True)

for i, j in lista_id:
    # Build the full path of the file
    ruta_archivo = os.path.join(carpeta_destino, f"empresa_sello_rsa_{i}_{j}.pdf")

    # Check if the file already exists
    if not os.path.exists(ruta_archivo):
        pdf_url = f"https://www.aragonempresa.com/empresas-sello-rsa/imprimir.php?idusuario={i}&idencuesta={j}"
        response = requests.get(pdf_url)
        with open(ruta_archivo, "wb") as file:
            file.write(response.content)
        print(f"Downloaded and saved {ruta_archivo}")
    else:
        print(f"File {ruta_archivo} already exists, skipping download.")
```

✓ 6m 55.4s Python

```
Downloaded and saved ..\data_ignore\EDA\empresa_sello_rsa_1000_7.pdf
Downloaded and saved ..\data_ignore\EDA\empresa_sello_rsa_1001_6.pdf
Downloaded and saved ..\data_ignore\EDA\empresa_sello_rsa_1002_6.pdf
Downloaded and saved ..\data_ignore\EDA\empresa_sello_rsa_1004_9.pdf
```

Extracción datos: Descarga formularios pdf, extracción datos y verificación

Negocio considera las siguientes variables relevantes y que son comunes a los 4 con matices que tendremos que tener en cuenta a la hora de determinar los patrones:

Variable	Tipo	Descripción
id_cliente	num	identificación del cliente
id_formulario	num.pdf	identifica los 4 tipos de formularios, revisar detecte correctamente los campos elegidos.
tipo_organizacion	object	Autonomo (120), Pequeña empresa (850), Mediana empresa (124), Gran empresa (113), Empresa publica (18), Entidades(301), Ong(47)
nombre_organizacion	object	Anonimizar, usaremos id_cliente
direccion	object	Convertir a latitud y longitus para geolocalizar en mapa
sector_actividad	object	Analizar, variable importante para agrupar, esta en lenguaje natural, interesante relacionar con nivel de contaminación Más_contaminantes/Normal/Menos_contaminantes(sectores ecologicos, gestión residuos)
year	object	Año comienzo actividad
persona_contacto	object	Nombre de la persona responsable. Anonimizar, se usara en la fase de contacto comercial
email_contacto	object	Email de contacto. Anonimizar, se usará en la fase de contacto comercial
pagina_web	object	Anonimizar. Solo util para profundizar departamento comercial para adecuar oferta de servicios y conocer al cliente.
impacto_actividad	object	Es lenguaje natural y recoge la valoración que da el cliente al impacto de su actividad en el medioambiente
prioridad mediambiental	num	la entidad prioriza la importancia que le otorga a los siguientes aspectos para lograr el éxito, negocio solo se centra en la importancia que se la da a "Trabajar para proteger y mejorar el medio ambiente" en la gestión socialmente responsable. (1 más importante - 10 menos importante) Opciones:
		Emplear prácticas empresariales abiertas y transparentes
		Gestionar con ejemplaridad
		Escuchar las necesidades y expectativas de sus grupos de interés
		Poner en práctica medidas responsables en la gestión de Recursos Humanos
		Ofrecer productos y servicios de calidad
		Capacitación y desarrollo profesional
		Disponer de presupuesto
		Trabajar para proteger y mejorar el medio ambiente
		Crear Programas con impacto positivo en la Comunicad Local
		Ser un negocio rentable y sostenible en el tiempo
mejora	object	A la pregunta del cuestionario ¿Podría enunciar un área de mejora? Es lenguaje natural y Negocio considera interesante su estudio para detectar necesidades en las entidad, entendemos que si no hay mejoras e. complicado aportar nuestros servicios

Extracción datos: Descarga formularios pdf, extracción datos y verificación

PyMuPDF is a Python module that is part of the PyMuPDF library, which is used to interact and work with PDF documents. It is especially useful for tasks related to PDF extraction, editing, and manipulation.

We use the following function to extract the selected data, keeping in mind that we have four forms and need to include different options.

```python
def extraer_datos(pdf_path):
    datos = {}

    with fitz.open(pdf_path) as pdf:
        texto_completo = ""

        # Concatenate all text from the PDF pages
        for page_num in range(len(pdf)):
            texto_completo += pdf[page_num].get_text()

        # Pattern to remove page skirts (ej. "[...] página X/XX")
        patron_faldon = r'''\[-\d+\]
        \s*RSA\s-\s
        (Entidades\sno\slucrativas|GRANDES\sEMPRESAS\sY\sEMPRESAS\sPÚBLICAS|AUTONOMOS-AS|PYMES)?
        \s*página\s\d+/\d+\n?
        '''
        # Remove footnotes from the entire text
        texto_sin_faldon = re.sub(patron_faldon, '', texto_completo, flags=re.VERBOSE | re.DOTALL)


        # ID del cliente y id_formulario
        datos["id_cliente"] = pdf_path.split("_")[-2]
        datos["id_formulario"] = pdf_path.split("_")[-1]

        # Tipo de organización
        patron_tipo = r"\n*RSA - (.*?)\n*Empresa evaluada"
        match_tipo = re.search(patron_tipo, texto_sin_faldon, re.DOTALL)
        if match_tipo:
            datos['tipo_organizacion'] = match_tipo.group(1).replace("\n", " ").strip()
```

Extracción datos: ejemplo formulario

Instituto Aragonés de Fomento

Departamento de Presidencia, Economía y Justicia

Calle Valenzuela, 9
50071 Zaragoza
Telf: 976 702 100
info@iaf.es

RSA - AUTONOMOS-AS

Empresa evaluada
INSIGHT MANAGEMENT SOLUTIONS
CIF:25457071G
C/Eduardo Jesús Taboada 8, 4A
50002 - ZARAGOZA
Zaragoza

DATOS DE LA EMPRESA

01 - Denominación de la organización

INSIGHT Management Solutions

02 - Dirección - CP - POBLACION

Calle de Pedro Cerbuna, 12, 50009 Zaragoza

03 - Sector. Actividad

Servicios profesionales. Consultoría de gestión.

04 - Año comienzo actividad.

2012

05 - Persona de contacto

José Ramón García Aranda

06 - e-mail contacto:

jrgarciaranda@gmail.com

07.- Página web

https://es.linkedin.com/in/jrgarciaranda

GENERAL

Tendencias Globales

1- Considera que su organización tiene en cuenta o se ve afectada, directa o indirectamente, por temas globales y generales como, por ejemplo:

- Económicos (por ejemplo, creación de empleo, generación de riqueza, cumplimiento de la legalidad)
- Políticos (por ejemplo, cambios de gobierno, alianzas con Administraciones Públicas)
- Medioambientales (cambio climático, consumo de recursos: agua, energía eléctrica, etc)
- Tecnológicos (por ejemplo, comunicaciones, redes sociales)
- Sociales (igualdad de género, conciliación de la vida personal y profesional, educación, formación, etc)

considerar que el objetivo de desplegar ese tipo de iniciativas no es obtener un mejor posicionamiento de "venta".

AMBIENTAL

Impacto ambiental

Instituto Aragonés de Fomento

Departamento de Presidencia, Economía y Justicia

17 - Respecto a los temas medioambientales, ¿cómo tiene en cuenta el impacto ambiental en el desarrollo de la actividad?

Se tiene en cuenta el impacto ambiental ocasionado por la actividad y se ha llevado a cabo alguna iniciativa de forma puntual e informal.

17.1. Información adicional.

Evidencias:
Si bien la actividad prestada (servicios profesionales) no suele generar un alto impacto medioambiental por sí misma, sí pueden tenerlo otros aspectos vinculados a la prestación en sí (Vg.: Viajes de larga duración). En este sentido, cuando, por ejemplo, es necesario realizar traslados/viajes de larga duración, se analizan los impactos medioambientales ocasionados por la actividad estableciendo medidas de actuación específicas tales como la concentración de actividades para minimizar desplazamientos y/o la puesta en común entre varios clientes.

Economía circular y comunicación

18 - ¿Conoce y aplica en su negocio los principios de economía circular?

Los principios de economía circular están integrados en el negocio, y se cuenta con una evaluación periódica de los resultados obtenidos.

18.1. Información adicional

Priorización de temas

20 - Priorice la importancia que le otorga a los siguientes aspectos para lograr el éxito en la gestión socialmente responsable. (1 más importante - 10 menos importante)

1 - Gestionar con ejemplaridad
2 - Escuchar las necesidades y expectativas de sus grupos de interés

3 - Ser un negocio rentable y sostenible en el tiempo
4 - Ofrecer productos y servicios de calidad
5 - Crear Programas con impacto positivo en la Comunicad Local
6 - Emplear prácticas empresariales abiertas y transparentes
7 - Trabajar para proteger y mejorar el medio ambiente
8 - Capacitación y desarrollo profesional
9 - Poner en práctica medidas responsables en la gestión de Recursos Humanos
10 - Disponer de presupuesto

Valoración Global

21 - ¿En qué nivel de 0 a 10 siendo 10 muy alto valora su satisfacción con la gestión socialmente responsable que actualmente realiza?

9

Puntos fuertes y áreas de Mejora

22 - ¿Podría enunciar un punto fuerte?

Hacer crecer a otros de manera sostenible forma parte del ADN de Insight Ma
Solutions.

23 - ¿Podría enunciar un área de mejora?

Poder crecer para desplegar de manera más "escalable" acciones

Agenda 2030 y Objetivos de Desarrollo Sostenible

¿Conoce la Agenda 2030 y los Objetivos de Desarrollo Soste

Sí, conozco la Agenda 2030 pero no he establecido ningún co
de los ODS.

**En caso afirmativo, ¿tiene identificados los ODS priorit
impacta?**

Si

Extracción datos: Descarga formularios pdf, extracción datos y verificación

To create our base dataframe we create a previous dictionary

```
diccionario_id = {usuario_id: encuesta_id for usuario_id, encuesta_id in lista_id}
```

[4] ✓ 0.0s

```
# Initialize the DataFrame with the appropriate columns and apply our extract function to
# each of the downloaded PDF forms:
df_base = pd.DataFrame()

keys = list(diccionario_id.keys())
values = list(diccionario_id.values())

for i in range(len(diccionario_id)):
    id_usuario = keys[i]
    id_formulario = values[i]
    pdf_path = f"..\\data_ignore\\EDA\\empresa_sello_rsa_{id_usuario}_{id_formulario}.pdf"
    datos = extraer_datos(pdf_path)
    df_base = pd.concat([df_base, pd.DataFrame([datos], index=[id_usuario])], ignore_index=False)
```

Extracción datos: Descarga formularios pdf, extracción datos y verificación

```
df_base.head()
```

Python

	id_cliente	id_formulario	tipo_organizacion	nombre_organizacion	direccion	sector_actividad	year	empleados	persona_contacto	email_contacto	pagina_web	impacto_actividad	prioridad mediambiental	mejora
1000	1000	7.pdf	GRANDES EMPRESAS Y EMPRESAS PÚBLICAS	EL CORTE INGLÉS S.A.	C/Hermosilla, nº 112 28009 Madrid	Sector comercio, grandes almacenes	1941	81.714 personas	Isabel Paricio Perales	isabel_paricio@elcorteingles.es	www.elcorteingles.es	Se evalúa el impacto ambiental y se ha desarro...	2	Implantación de programas de comunicación global.
1001	1001	6.pdf	PYMES	AYANET RRHH, S.L.	Calle Bari 57, Edificio TIC XXI (Plaza) 50.197...	Consultoría integral de Recursos Humanos: - Se...	1985	8	Tania Grande Maza	tgrande@ayanet.es	www.ayanet.es	Se evalúa el impacto ambiental y se llevan a c...	10	Quizá deberiamos trabajar algo más en materia ...
1002	1002	6.pdf	PYMES	IMPROVING, CONSULTORIA Y FORMACION, S.L. (IMFO...	AVENIDA DE LA JACETANIA, 21, 22700 JACA HUESCA	FORMACION PARA EL EMPLEO	2010	7	ISABEL VITALLÉ ZAURIN	IVITALLE@EIMPROVING.ES	WWW.IMFORMA.ES	Se evalúa el impacto ambiental y se ha desarro...	4	Mejorar en materiales didácticos e indicadores...
1004	1004	9.pdf	Entidades no lucrativas	ACISJF IN VIA	Paseo Echegaray y Caballero, 118, 50002 (Zarag...	Mujer/Acción Social	En Zaragoza en 1953	8	María Jesús Soler Cochi. Directora de Programas	acisjfzaragoza@gmail.com	www.acisjfzaragoza.org	Se evalúa el impacto ambiental y se llevan a c...	7	Dar estabilidad al área económica, con proyect...
1005	1005	6.pdf	PYMES	ELECTRICIDAD AMARO, S.A.	Carretera del aeropuerto, km. 5,300 50.190 - Z...	Instalaciones eléctricas en general.	1982	4	Óscar Amaro Arceiz	oscar@eamaro.com	www.eamaro.com	Se tiene en cuenta el impacto ambiental ocasio...	6	Desarrollar más acciones sociales. Debemos mej...

```
# Guardamos el dataframe en un archivo csv en la carpeta data_sample
df_base.to_csv(r"..\data_sample\datos_rsa.csv")
```

3.3 Cargamos nuestro archivo base guardado.

```
df_base = pd.read_csv("../data_sample/datos_rsa.csv")
```

Limpieza df_base

Variable	Type	Description	Action
Unnamed:0	int	Automatic, duplicate of id_cliente	Delete
id_cliente	num	Client identification	Keep, base for anonymization
id_formulario	object	Form type, only used for printing	Not useful
tipo_organizacion	object	Self-employed (120), Small business (850), Medium business (124), Large company (113), Public company (18), Entities (301), NGOs (47)	Keep
nombre_organizacion	object	Client name	Anonymize, we will use id_cliente
direccion	object	Client address	Convert to latitude and longitude for geolocation on a map
sector_actividad	object	Activity sector	Analyze, important variable for grouping, it is in natural language, it is interesting to relate it to pollution level: Most_polluting/Normal/Less_polluting (ecological sectors, waste management)
year	object	Year of creation	Not considered interesting to convert to datetime, review because there may be different formats yyyy, dd/mm/yyyy...
empleados	object	Number of employees	In natural language, needs to be reviewed
persona_contacto	object	Name of the responsible person	Anonymize, will be used in the commercial contact phase
email_contacto	object	Contact email	Anonymize, will be used in the commercial contact phase
pagina_web	object	Client website	Anonymize. Only useful for deepening the commercial department to tailor the service offer and get to know the client. Something is missing
impacto_actividad	object	It is natural language and reflects the client's assessment of the environmental impact of their activity	Count relevant words as an indicator of significant environmental activity; a higher number suggests a greater environmental impact
prioridad mediambiental	float	The entity prioritizes the importance given to "Working to protect and improve the environment" in socially responsible management (1 most important - 10 least important)	Convert to int
mejora	object	To the questionnaire question "Could you state an area for improvement?" It is in natural language and the business considers it interesting to study to detect needs in the entity, understanding that if no improvements are detected, it is likely that the company is less open to hiring our services	Count relevant words as an indicator of the need for advice; if the company does not identify areas for improvement, it is less open to our services.

Limpieza df_base: coordenadas

4.1 Converting address to coordinates

```python
# We use OpenCage to locate the latitude and longitude coordinates of the addresses
# We define a function that first checks if we already have the coordinates in our JSON notebook.
# If we have coordinates, it takes them from the notebook; otherwise, it uses the API.
# If there is no address, it tries the postal code, and as a last resort, it tells us the records that
# either it has not located or do not have the postal code.


#key = 'c8631658aa1d4f688649d97490bfdb23'
key = '868719c58b7843de8591665f0e53848f'
geocoder = OpenCageGeocode(key)

# File path where the results are saved
cache_file = '../utils/geocoding_cache.json'

def geocode_addresses(df, key, cache_file):
    geocoder = OpenCageGeocode(key)

    # Load existing cache
    if os.path.exists(cache_file):
        with open(cache_file, "r") as f:
            cache = json.load(f)
    else:
        cache = {}

    # Initialize latitude and longitude columns
    df["latitud"] = None
    df["longitud"] = None

    for i in df_limpio.index:
        try:
            # Check if the address is valid
            query = str(df.loc[i, "direccion"])
            if pd.isnull(query) or query.strip() == "":
                print(f"Dirección vacía para id_cliente: {df.loc[i, 'id_cliente']}")
                continue
```

```python
            # Realizar consulta a OpenCage
            results = geocoder.geocode(query)
            if results:
                lat = results[0]["geometry"]["lat"]
                lng = results[0]["geometry"]["lng"]
                df.at[i, "latitud"] = lat
                df.at[i, "longitud"] = lng
                cache[query] = {"lat": lat, "lng": lng}
                continue

            # Si no hay resultados, intentar con el código postal
            match = re.search(r"\b(50\d{3}|22\d{3}|44\d{3})\b", query)
            if match:
                postal_code = match.group(0)
                query_postal = f"{postal_code}, Spain"

                if query_postal in cache:
                    df.at[i, "latitud"] = cache[query_postal]["lat"]
                    df.at[i, "longitud"] = cache[query_postal]["lng"]
                else:
                    results_postal = geocoder.geocode(query_postal)
                    if results_postal:
                        lat = results_postal[0]["geometry"]["lat"]
                        lng = results_postal[0]["geometry"]["lng"]
                        df.at[i, "latitud"] = lat
                        df.at[i, "longitud"] = lng
                        cache[query_postal] = {"lat": lat, "lng": lng}
                    else:
                        print(f"No se encontraron coordenadas: {df.loc[i,
            else:
                print(f"Dirección sin código postal: {df.loc[i, 'id_client

            # Pausa entre consultas para evitar límites de velocidad
            time.sleep(1)

        except Exception as e:
            print(f"Error procesando id_cliente {df.loc[i, 'id_cliente']}:

    # Guardar caché actualizada
    with open(cache_file, "w") as f:
        json.dump(cache, f, indent=4)

    return df
```

```python
# Diccionario con los id_cliente y sus códigos postales corregidos manualmente
codigos_postales_correcciones = {
    1226: "50600",
    1601: "50001",
    1645: "50010",
    1674: "50001",
    1744: "22002",
    2100: "50001",
    2209: "22700",
    2397: "50001",
    253: "50003",
    2680: "50016",
    2683: "50001",
    2759: "50008",
    2831: "50171",
    3032: "50001",
    3087: "44600",
    494: "50009",
    585: "22197",
    623: "44556",
    840: "50001",
    943: "50014",
    952: "50001"
}

# Añadir manualmente los códigos postales al DataFrame
for id_cliente, codigo_postal in codigos_postales_correcciones.items(
    # Localizar la fila por el id_cliente y asignar el código postal m
    if id_cliente in df_limpio["id_cliente"].values:
        df_limpio.loc[df_limpio["id_cliente"] == id_cliente, "direccion

# Verificar las filas actualizadas
print(df_limpio[df_limpio["id_cliente"].isin(codigos_postales_correcciones.
```

Limpieza df_base: year

```python
# Function to detect the first valid year in different formats
# Function to clean periods and extract the first 4-digit year
def detectar_year(texto):
    if not texto or not isinstance(texto, str):
        return None

    texto_limpio = re.sub(r'(\d+)\.(\d+)', r'\1\2', texto)

    patron_year = r'\b(15\d{2}|16\d{2}|17\d{2}|18\d{2}|19\d{2}|20\d{2})\b'
    match = re.search(patron_year, texto_limpio)
    if match:
        return match.group(1)
    return None
```

✓ 0.0s

```python
df_limpio["year_2"] = df_limpio["year"].apply(detectar_year)
```

✓ 0.0s

```python
# We analyze the unique values to detect if there are nulls:
valores_unicos = df_limpio["year_2"].astype(str).unique()
valores_erroneos = [y for y in valores_unicos if not re.fullmatch(r"\d{4}", str(y))]
valores_unicos
```

✓ 0.0s

```
array(['1941', '1985', '2010', '1953', '1982', '1999', '1910', '1990',
       'None', '2018', '1958', '1975', '2016', '1995', '2009', '1964',
       '1997', '1977', '1993', '1996', '2015', '2013', '1944', '1962',
       '2004', '1991', '2014', '1989', '2006', '2005', '2003', '2001',
       '1983', '2000', '2017', '2011', '1998', '2007', '1981', '2012',
       '1966', '1929', '2008', '1961', '1984', '1965', '1952', '1946',
       '2002', '1992', '1956', '1986', '1979', '1954', '1967', '1974',
       '1980', '1963', '1968', '1978', '1723', '1925', '1969', '1876',
       '1988', '1927', '1987', '1899', '1905', '2019', '1994', '1931',
       '1912', '1900', '1948', '1934', '2023', '1886', '1971', '1957',
       '2020', '1960', '1945', '1903', '1950', '1935', '1972', '1940',
       '2022', '1973', '1874', '1959', '1943', '1947', '2021', '1955',
```

```python
# We created a manual dictionary with the date corrections for the 27 records
fechas_correcciones = {1014: '1983',
 1051: '2005',
 1156: '1980',
 1230: '1994',
 136: '1996',
 1446: '1960',
 1457: '1960',
 1525: '1955',
 1592: '1976',
 169: '1998',
 1955: None,
 2391: '1990',
 2874: '1994',
 3236: '2014',
 3324: '2019',
 3338: '1994',
 3417: '1948',
 346: '2007',
 3522: '2020',
 465: '1969',
 542: '1975',
 548: '1992',
 660: '1962',
 696: None,
 960: '2000'}

for id_cliente, fecha in fechas_correcciones.items():

    if id_cliente in df_limpio["id_cliente"].values:
        df_limpio.loc[df_limpio["id_cliente"] == id_cliente, "year_2"] =


print(df_limpio[df_limpio["id_cliente"].isin(fechas_correcciones.keys())][[
```

✓ 0.0s

Limpieza df_base: empleados

4.3 Analysis of the variable: empleados

We encounter numeric values and text, it will be necessary to convert to numeric and check for errors.

```python
# Función para detectar trabajadores
def detectar_trabajadores(texto):
    if not texto or not isinstance(texto, str):  # Validar que el valor no sea nulo o no sea cadena
        return None
    # Quitar puntos de los números, como transformar "1.996" en "1996"
    texto_limpio = re.sub(r'(\d+)\.(\d+)', r'\1\2', texto)
    # Patrón para encontrar el primer numero
    patron_trabajadores = r'\b(\d+)\b'
    match = re.search(patron_trabajadores, texto_limpio)
    # if match:
    #     return match.group(1)
    # return None
    numeros_encontrados = re.findall(patron_trabajadores, texto_limpio)
    return numeros_encontrados
```

✓ 0.0s

```python
# Crear la columna 'empleados_2' si no existe
if 'empleados_2' not in df_limpio.columns:
    df_limpio['empleados_2'] = None

for i in range(len(df_limpio)):
    if df_limpio.iloc[i]["tipo_organizacion"] == "AUTONOMOS-AS":
        df_limpio.iloc[i, df_limpio.columns.get_loc("empleados_2")] = 1
    else:
        # Detectar el número de trabajadores usando la función y asignarlo a empleados_2
        trabajadores = detectar_trabajadores(df_limpio.iloc[i]["empleados"])
        if trabajadores:
            df_limpio.iloc[i, df_limpio.columns.get_loc("empleados_2")] = trabajadores[0]
        else:
            df_limpio.iloc[i, df_limpio.columns.get_loc("empleados_2")] = None
```

✓ 0.2s

```python
# Diccionario con los id_cliente y empleados corregidos manualmente, si no se aporta y no se valorara relevante se pone a 0
empleados_correcciones = {1014: '3',
 1054: '0',
 1171: '2',
 1286: '3',
 1326: '6',
 1343: '0',
 1344: '1',
 1348: '2',
 1357: '5',
 1461: '0',
 1462: '0',
 1495: '3',
461: '0',
508: '0',
559: '11',
580: '20',
639: '2',
753: '0',
906: '4',
926: '5'}

# Añadir manualmente los empleados al DataFrame, la mayoria datos en letra
for id_cliente, empleados in empleados_correcciones.items():
    # Localizar la fila por el id_cliente y asignar el código postal manualmente
    if id_cliente in df_limpio["id_cliente"].values:
        df_limpio.loc[df_limpio["id_cliente"] == id_cliente, "empleados_2"] = empleados

# Verificar las filas actualizadas
print(df_limpio[df_limpio["id_cliente"].isin(empleados_correcciones.keys())][["id
```

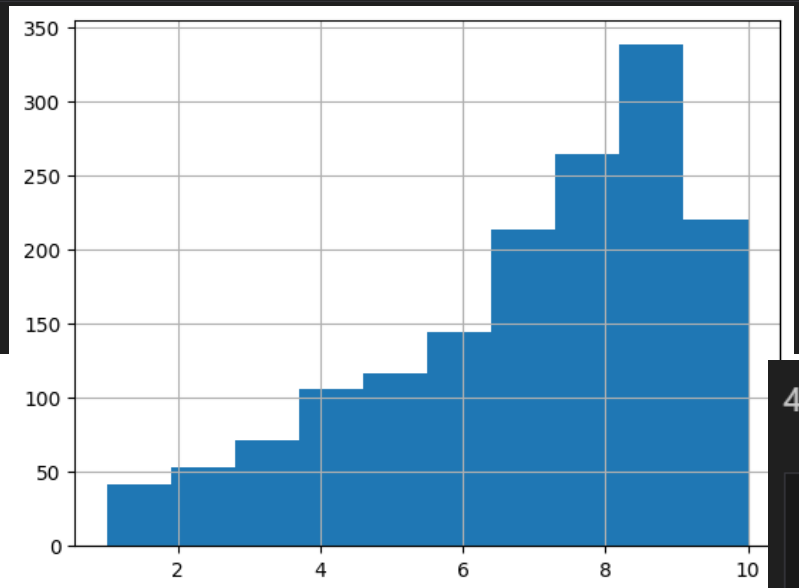
Limpieza df_base: prioridad mediambiental y mejora

4.4 Analisys of the variable: prioridad mediambiental

```
df_limpio["prioridad mediambiental"].value_counts().sort_values()

#Prioritize the importance you give to the following aspects for achieving success in socially responsible management. (1 most important - 10 least important)
#Work to protect and improve the environment
```

0.0s

```
prioridad mediambiental
1.0      42
2.0      53
3.0      71
4.0     106
5.0     116
6.0     144
7.0     213
10.0    220
8.0     264
9.0     338
Name: count, dtype: int64
```



4.5 Analisys os the variable: mejora

```
# There are null records, in these cases we will mark it as "no
sin_mejora = "sin_mejora"
df_limpio.loc[:, 'mejora'] = df_limpio['mejora'].fillna(sin_
```

0.0s

Limpieza df_base: sector_actividad

4.6 Analisys of teh variable: sector actividad

```
# Find the nulls:
df_limpio[df_limpio["sector_actividad"].isnull()]
```

✓ 0.0s

	id_cliente	id_formulario	tipo_organizacion	nombre_organizacion	direccion	sector_actividad	year	empleados
297	1584	9.pdf	Entidades no lucrativas	Feria de Zaragoza	Autovía A II, Km. 311 50012 Zaragoza	NaN	1945	61
722	2508	6.pdf	PYMES	ELECTRICIDAD MAYASA S.L.	Polígono Miguel Servet, nave 3. 50013 Zaragoza	NaN	En 1967, primero como persona física y más tarde como sociedad limitada.	24

2 rows × 22 columns

```
# Dictionary with the client_ids and their sectors of activity manually corrected
codigos_sector_actividad = {
    1584: "Organización de convenciones y ferias de muestras.CNAE: 8230",
    2508: "Instalaciones eléctricas de BAJA TENSIÓN-- Instalaciones especiales- Revisión, reparación y bobina
}
```

Variables tratamiento lenguaje natural:

5. Processing variables with natural language.

We still need to process the variables with natural language.

We are going to apply tokenization processes and clean up unnecessary characters.

In the case of **"impacto_actividad"**, Business assesses that the greater the number of words, the greater the impact of the activity on the environment.

The same applies to **"mejora"**: the greater the number of words, the greater the attractiveness of the entity to offer our services.

However, with **"sector_actividad"**, which has a high degree of cardinality due to the lack of a clear limiting pattern, we have tried different embedding models without consistent results. It is considered a relevant variable, and therefore, a manual classification of the activity sectors into three groups is carried out by business:

- High-level polluting activities (1)
- Medium-level polluting activities (0)
- Low-level polluting activities (-1) such as waste management, environmental...

```python
# Natural language object variable to clean are:
variables_ln = ["sector_actividad", "impacto_actividad", "mejora"]
```

✓ 0.0s Python

Variables tratamiento lenguaje natural:

```python
# Expresiones regulares para reemplazar caracteres
REPLACE_NO_SPACE = re.compile(r"(\.)|(\;)|(\:)|(\!)|(\?)|(\,)|(\")|(\()|(\))|(\[)(\])|(\d+)")
REPLACE_WITH_SPACE = re.compile(r"(<br \s*/><br\s*/>)|(\-)|(\/)|(_)")

NO_SPACE = ""
SPACE = " "

# Función para eliminar acentos
def remove_accents(input_str):
    nfkd_form = unicodedata.normalize('NFKD', input_str)
    return ''.join([char for char in nfkd_form if not unicodedata.combining(char)])

# Función para limpiar texto
def clean(row):
    # Paso 1: Eliminar acentos
    row = remove_accents(row)

    # Paso 2: Excluir el patrón de CNAE seguido de números antes de limpiar
    match_cnae = re.findall(r'CNAE[:\s]*\d+', row)

    # Limpio signos y convierto a minúsculas
    dato = REPLACE_NO_SPACE.sub(NO_SPACE, row.lower())

    # Convierto retornos de carro y guiones en espacios
    dato = REPLACE_WITH_SPACE.sub(SPACE, dato)

    # Quito cualquier link
    dato = " ".join([word for word in dato.split() if "http" not in word])

    # Recupero "CNAE: ####" al texto limpio
    dato = " ".join(match_cnae + dato.split())

    return dato
```

```python
for i in variables_ln:
    df_limpio[i+"_clean"] = df_limpio[i].apply(clean)
```

```python
variables_ln_clean = ["sector_actividad_clean", "impacto_actividad_clean", "mejora_clean"]
```

```python
import nltk
nltk.download('popular')
```

```python
dictionary = stopwords.words("spanish")

def remove_stopwords(row):
    # Expresión regular para detectar "CNAE" seguido de números
    pattern = r'CNAE \d+'

    # Divide las palabras y aplica la lógica
    dato = " ".join([word for word in row.split(" ")
                     if word not in dictionary or re.match(pattern, word)])
    return dato
```

```python
for i in variables_ln_clean:
    df_limpio[i+"_sin_stopwords"] = df_limpio[i].apply(remove_stopwords)
```

```python
df_limpio["impacto_recuento"] = df_limpio["impacto_actividad_clean_sin_stopword
df_limpio["mejora_recuento"] = df_limpio["mejora_clean_sin_stopwords"].apply(l
```

Variables tratamiento lenguaje natural:

```
# Get unique values from the column

actividades = df_limpio["sector_actividad_clean_sin_stopwords"].value_counts()
```
✓ 0.0s

```
df_actividades = actividades.reset_index()
df_actividades.columns = ['sector_actividad', 'count']
```
✓ 0.0s

```
# We save the dataframe in a csv file in the data_sample folder to make it easier to fill in Excel
df_actividades.to_csv(r"..\data_sample\datos_actividades.csv", index=True)
```
✓ 0.0s

```
# We call the manually filled excel
df_actividades_manual = pd.read_excel(r"..\data_sample\clasificacion.xlsx", usecols=["sector_actividad", "clasificacion_manual"])
```
✓ 0.0s

```
# Create a classification dictionary from df_activities_manual
diccionario_clasificaciones = dict(zip(df_actividades_manual['sector_actividad'], df_actividades_manual['clasificacion_manual']))


# Add the classification to the clean_df using map

df_limpio['clasificacion'] = df_limpio['sector_actividad_clean_sin_stopwords'].map(lambda x: diccionario_clasificaciones.get(x, 0))
```
✓ 0.0s

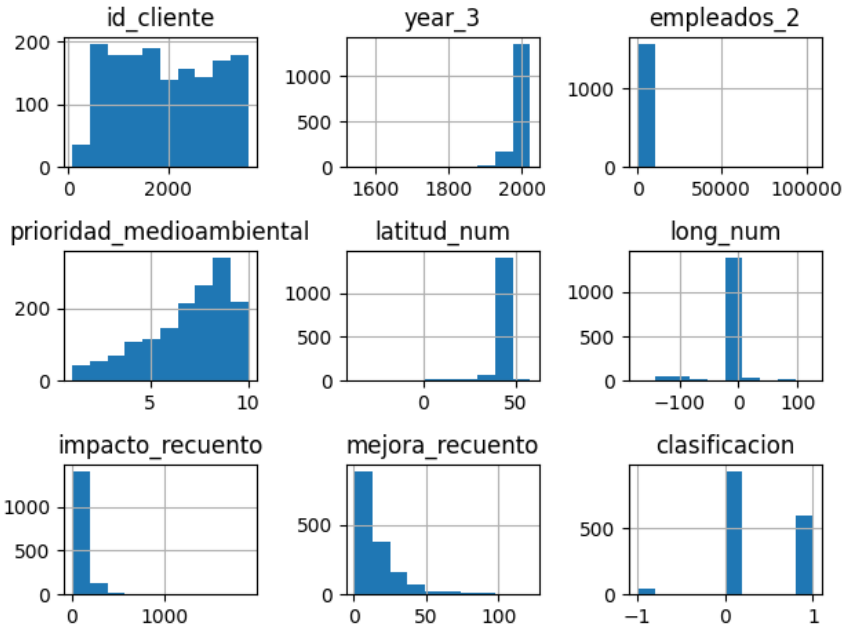
Mini eda: df_clasificado

```
describe_df(df_clasificado).T
```
✓ 0.0s

COL_N	DATA_TYPE	MISSINGS (%)	UNIQUE_VALUES	CARDIN (%)
id_cliente	int64	0.0	1570	1.0
tipo_organizacion	object	0.0	4	0.0
year_3	int32	0.0	118	0.08
empleados_2	Int64	0.0	237	0.15
prioridad_medioambiental	int32	0.0	10	0.01
latitud_num	float64	0.0	770	0.49
long_num	float64	0.0	769	0.49
impacto_recuento	int64	0.0	311	0.2
mejora_recuento	int64	0.0	93	0.06
clasificacion	int64	0.0	3	0.0

Variable	Tipo	Descripción
id_cliente	int	Numero de entidad(1570 entidades)
tipo_organizacion	object	4 tipos de entidades (autonomos,gran empresa, publicas, entidades no lucrativas)
year_3	int	Año de creacion de la entidad
empleados_2	int	Numero de empleados
prioridad mediambiental	int	Escala de 1 a 10 la importancia que le da a medioambiente, tal vez reducir a 3 para mejorar la interpretacion de resultados
latitud_num y long_num	float	coordenadas de la direccion
impacto_recuento	int	312 valores, mas valor mas impacto medioambiental
mejora_recuento	int	93 valores, mas valor mas interesante el cliente
clasificacion	int	1,0,-1 (Muy contaminante, Medio, Bajo contaminante)

Mini eda: df_clasificado



We apply the scaling by consulting with the business, which tells us that since all the entities are located in Aragon, classification by coordinates is discarded

```python
# Prioridad medioambiental reducimos de 10 a 2 tipos
def mapping_prioridad(x):
    if x > 5:
        return 1 # Entidades que NO TIENEN en su top 5 la prioridad mediambiental
    else:
        return 0 # Entidades que lo considerar en su top 5

df_clasificado.loc[:, 'prioridad_medioambiental_map'] = df_clasificado['prioridad_medioambiental'].apply(mapping_prioridad)
```

✓ 0.0s

```python
# Normalizamos la variable year la reducimos a 4 escalas para simplificar
def mapping_year(x):
    if pd.isna(x):
        return 3
    elif x > 2020:
        return 3
    elif x > 2000:
        return 2
    elif x > 1980:
        return 1
    else:
        return 0

df_clasificado.loc[:, 'year_map'] = df_clasificado['year_3'].apply(mapping_year)
```

✓ 0.0s

Arquitectura solución:

```
#  Nuestras variables finales:

features_cat = ["tipo_organizacion", "prioridad_medioambiental_map", "clasificacion"]
features_num = ["year_map", "empleados_map","impacto_recuento_map", "mejora_recuento_map"]
```

7. Solution architecture

```
train_set = df_clasificado[features_cat+features_num]
```

✓ 0.0s

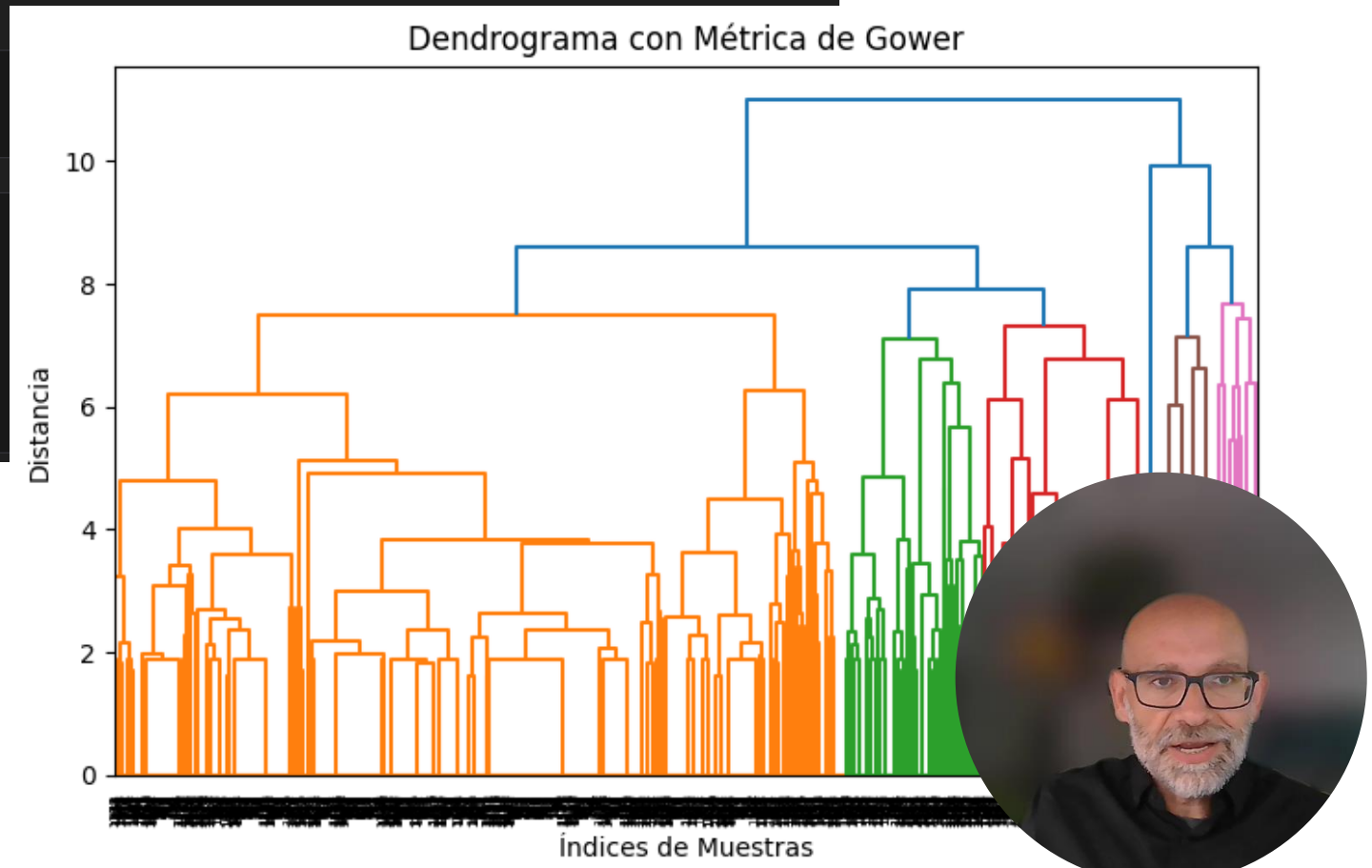
Gower distance is useful when working with data sets that include different types of variables, such as categorical, numerical, and ordinal variables. Unlike other distance metrics (such as Euclidean or Manhattan), Gower is designed to handle this type of heterogeneous data efficiently.

```
# Calcular la matriz de Gower

gower_matrix = gower.gower_matrix(train_set)
```

✓ 0.7s

```
# Generar el enlace jerárquico utilizando la matriz de Gower
linked = linkage(gower_matrix, method='average')

# Crear el dendrograma
plt.figure(figsize=(8, 5))
dendrogram(linked, labels=train_set.index.tolist(), distance_sort='ascending')
plt.title("Dendrograma con Métrica de Gower")
plt.xlabel("Índices de Muestras")
plt.ylabel("Distancia")
plt.show()
```

✓ 3.2s



Arquitectura solución: Selección Cluster objetivo

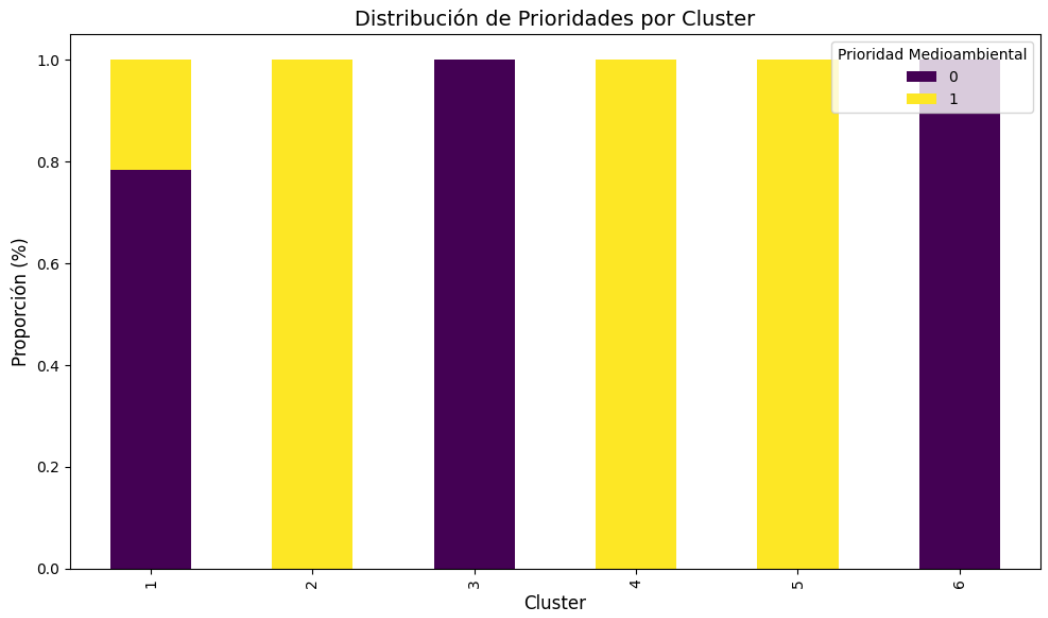
We have 6 clusters identified by the model, the next step will be to choose the one requested for Business

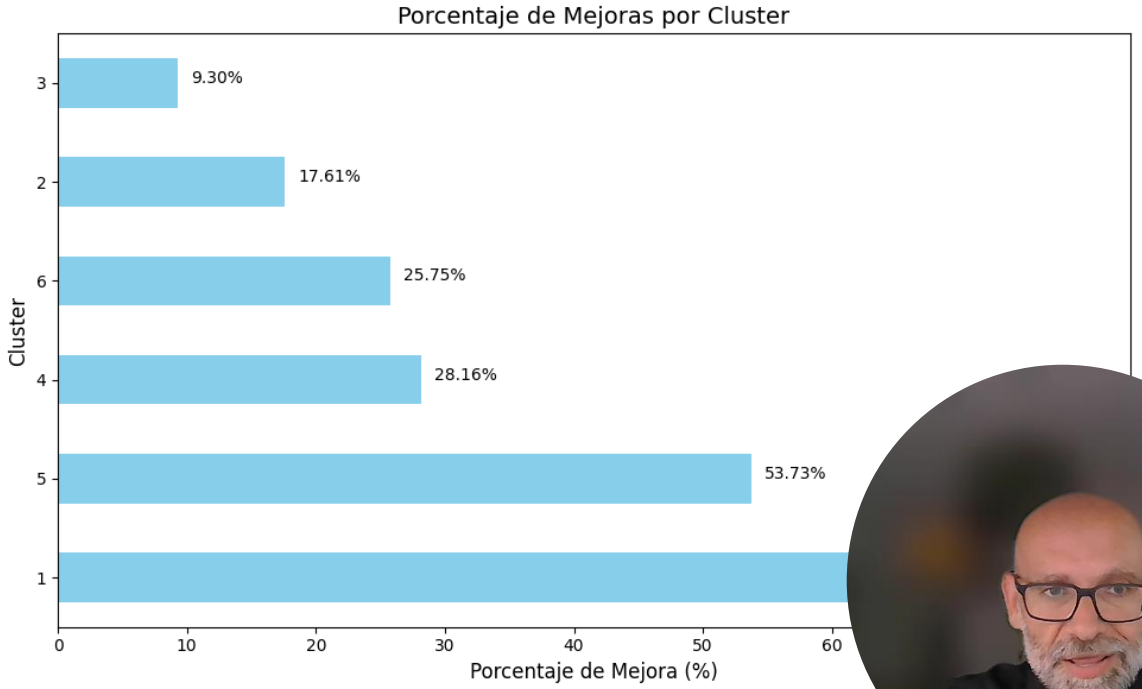
```
# Clustering jerárquico basado en la matriz de distancia de Gower
clusters = linkage(gower_matrix, method='complete')

# Obtener las etiquetas de los clusters
labels = fcluster(clusters, t=6, criterion='maxclust')

train_set.loc[:, 'Cluster'] = labels
```

```
train_set["Cluster"].value_counts()
```

✓ 0.0s

```
Cluster
2    800
4    243
3    235
5    126
6    106
1     60
Name: count, dtype: int64
```





Arquitectura solución: Selección Cluster objetivo. Cluster 6.

We propose our Cluster 6 to Business because these are entities that have the environment among their top five priorities and are clearly interested in improvements, as 25.75% of the entities in this group exceeded the 25-word threshold on their form.

We will send Business the list of the 106 entities with their contact information for the marketing department:

markdown

```
df_cluster = pd.concat([train_set[["Cluster"]],
                        df_clasificado[["id_cliente"]]], axis=1)
```

✓ 0.0s Python

```
df_cluster_6 = df_cluster[df_cluster["Cluster"]==6]
len(df_cluster_6)
```

✓ 0.0s Python

```
106
```

```
merged_df = pd.merge(df_cluster_6, df_limpio, on='id_cliente', how='left')
merged_df.sample().T
```

✓ 0.0s Python

	59
Cluster	6
id_cliente	3104
id_formulario	9.pdf
tipo_organizacion	Entidades no lucrativas
nombre_organizacion	ASOCIACION ARTRITIS OSCENSE (ARO)
direccion	C/ Berenguer 2-4,4ª Planta 22002 Huesca
sector_actividad	Provincia de Huesca
year	2010

Modelo preselección: df_supervisado

8. Ampliación a supervisado

Once our business cluster has been validated, we are faced with the challenge of leveraging this information to create a model that, based on external company data, can infer whether the company has an environmental priority. In these cases, we do not have any type of survey.

In this case, we propose a model that, based on variables such as organization type, activity classification, year of creation, and number of employees, predicts which company fits into our target cluster.

This can be a useful marketing tool for selecting companies.

```
# Seleccionamos las variables externas sin mapeo:

df_supervisado = pd.concat([train_set[["tipo_organizacion", "clasificacion", "Cluster"]],
                            df_clasificado[["year_3", "empleados_2"]]], axis=1)
```

✓ 0.0s

```
# Simplificamos a nuestro cluster objetivo 6

def mapping_cluster(x):
    if x == 6:
        return 1 # Si es del cluster 6
    else:
        return 0

df_supervisado.loc[:, 'cluster_obj'] = df_supervisado['Cluster'].apply(mapping_cluster)
```

✓ 0.0s

```
df_supervisado = df_supervisado.drop(columns= "Cluster")
```

✓ 0.0s

```
train, test = train_test_split(df_supervisado, test_size=0.2, random_state=42)
```

Modelo preselección: Definición

```
# Definición del problema de clasificación.

# target_clf → Variable objetivo para clasificación.
#     - Se ha definido previamente como "cluster_obj".
#     - 1 si esta en nuestra cluster 5, 0 en caso contrario.
#     - Es un problema de clasificación **binario** (2 categorías posibles).

target_clf = "cluster_obj"

y_train_cat = train["cluster_obj"]
y_test_cat = test["cluster_obj"]
```

```
# Definición de las características (features) utilizadas en la clasificación:

# features_cat_clf → Variables categóricas
#     - Contiene solo la variable "tipo_organizacion" ().
#     - Se debe transformar con OneHotEncoder para convertirla en variables numéricas.

# features_num_clf_1 → Variables numéricas
#     - clasificacion ya esta normalizada: 1 contaminante 0 media -1 ecologica
# features_num_clf_2
#     - year_3 le aplicaremos ordinal_encoding y la consideraremos como categorica
#     - empleados_2 le aplicaremos orinal encoding y se considera categorica


features_cat_clf = ["tipo_organizacion","clasificacion"]
features_num_clf = ["year_3", "empleados_2"]
```

```
# Definición de columnas a incluir y excluir en el modelo de clasificación.

# columns_to_keep_clf → Columnas que se mantendrán en el modelo de clasificación.
#     - Incluye:
#       - features_num_clf_1 (variables numéricas relevantes).
#       - features_cat_clf (variables categóricas a transformar con OneHotEncoder).

# columns_to_exclude_clf → Columnas que se excluirán del modelo de clasificación.
#     - Se obtienen eliminando de df.columns las variables incluidas en columns_to_keep_clf.
#     - Estas columnas no serán utilizadas en el modelo.

columns_to_keep_clf =  features_num_clf + features_cat_clf

columns_to_exclude_clf = [col for col in train.columns if col not in columns_to_keep_clf]

columns_to_exclude_clf
```

```
['cluster_obj']
```

Modelo preselección: pipeline preprocesamiento

```
# Definición de Pipelines para preprocesamiento de datos en clasificación.

# cat_pipeline → Preprocesamiento de variables categóricas.
#     - "Impute_Mode": Imputa valores faltantes con la moda (valor más frecuente).
#     - "OHEncoder": Aplica OneHotEncoder, ignorando categorías desconocidas, en lugar generar un error

# logaritmica → Transformación logarítmica de variables numéricas.
#     - Usa FunctionTransformer con np.log1p para estabilizar distribuciones sesgadas.
#     - feature_names_out="one-to-one" mantiene los nombres originales de las características.

# num_pipeline → Preprocesamiento de variables numéricas.
#     - "Impute_Mean": Imputa valores faltantes con la media.
#     - "logaritmo": Aplica la transformación logarítmica definida antes.
#     - "SScaler": Aplica StandardScaler para normalizar las variables numéricas.

# imputer_step_cat → ColumnTransformer para aplicar los Pipelines según el tipo de variable.
#     - "Process_Numeric": Aplica num_pipeline a features_num_clf_1 (variables numéricas).
#     - "Process_Categorical": Aplica cat_pipeline a features_cat_clf (variables categóricas).
#     - "Exclude": Elimina las columnas en columns_to_exclude_clf.
#     - remainder="passthrough": Mantiene cualquier otra columna sin modificar.

# pipe_missings_cat → Pipeline final que aplica el ColumnTransformer imputer_step_cat.
```

```
cat_pipeline = Pipeline(
    [("Impute_Mode", SimpleImputer(strategy="most_frequent")),
     ("OHEncoder", OneHotEncoder(handle_unknown='ignore'))
    ]
)
logaritmica = FunctionTransformer(np.log1p, feature_names_out="one-to-one")
# Convierte las funciones a transformadores
# mapping_empleados_transformer = FunctionTransformer(mapping_empleados)
# mapping_year_transformer = FunctionTransformer(mapping_year)

num_pipeline = Pipeline(
    [("Impute_Mean", SimpleImputer(strategy = "mean")),
     ("logaritmo", logaritmica),
     ("SScaler", StandardScaler()),
    ]
)

imputer_step_cat = ColumnTransformer(
    [("Process_Numeric", num_pipeline,features_num_clf)
     ("Process_Categorical", cat_pipeline, features_ca
     ("Exclude", "drop", columns_to_exclude_clf)],
       remainder = "passthrough"
    )

pipe_missings_cat = Pipeline([("first_stage", imputer_s
```

Modelo preselección: pipeline modelo clasificación

```
# Definición de Pipelines para modelos de clasificación.
#
# - Se crean pipelines que combinan preprocesamiento y modelos de clasificación.
# - Cada pipeline aplica primero `pipe_missings_cat`, que maneja datos faltantes y
# codificación de variables categóricas.
# - Luego, se entrena un modelo de clasificación diferente en cada pipeline:
#     - LogisticRegression (Regresión Logística)
#     - RandomForestClassifier (Bosques Aleatorios)
#     - LGBMClassifier (LightGBM)
#
# - Finalmente, se evalúan los modelos con validación cruzada usando "recall" como métrica.
```

```
logistic: 0.6500
[0.64705882 0.875      0.375      0.82352941 0.52941176]
randomF: 0.0000
[0. 0. 0. 0. 0.]
LGBM: 0.1096
[0.11764706 0.3125     0.         0.05882353 0.05882353]
```

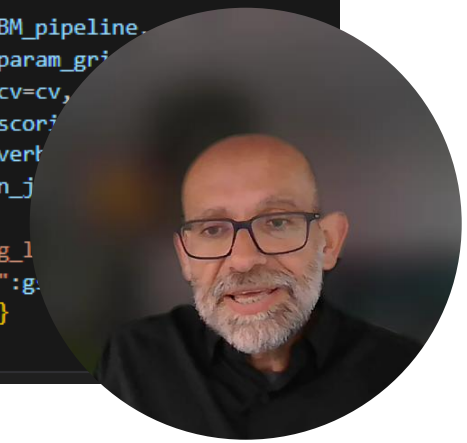
```python
# Pipeline con Regresión Logística
logistic_pipeline = Pipeline(
    [("Preprocesado", pipe_missings_cat),  # Paso de preprocesamiento
     ("Modelo", LogisticRegression(max_iter=10000, class_weight="balanced"))
    ])

# Pipeline con RandomForestClassifier
random_pipeline = Pipeline(
    [("Preprocesado", pipe_missings_cat),
     ("Modelo", RandomForestClassifier(class_weight="balanced"))
    ])


# Pipeline con LightGBMClassifier
LGBM_pipeline = Pipeline(
    [("Preprocesado", pipe_missings_cat),
     ("Modelo", LGBMClassifier(verbose=-1, class_weight="balanced"))
    ])

# Evaluación de los modelos con validación cruzada
for name, pipe in zip(["logistic","randomF","LGBM"],
                      [logistic_pipeline, random_pipeline, LGBM_pipeline]):
    resultado = cross_val_score(pipe, train, y_train_cat, cv=5, scoring="r

    # Mostrar el desempeño de cada modelo
    print(f"{name}: {np.mean(resultado):.4f}")
    print(resultado)
```

Modelo preselección: evaluación hiperparámetros

```
# Evaluación de hiperparámetros para modelos de clasificación con GridSearchCV.
#
# - Se definen diferentes conjuntos de hiperparámetros para los modelos:
#   - Regresión Logística (LogisticRegression)
#   - Random Forest (RandomForestClassifier)
#   - LightGBM (LGBMClassifier)
#
# - Se aplica GridSearchCV para encontrar la mejor combinación de hiperparámetros.
# - Se usa validación cruzada con 5 particiones (cv=5) y "recall" como métrica de
#       evaluación. Razón: Si tu objetivo es captar a todos los clientes relevantes
#       (por ejemplo, clientes con alta probabilidad de compra o retención),
#       es más importante minimizar los falsos negativos.
#       Esto significa que estarás dispuesto a aceptar algunos falsos positivos.
# - Se almacena cada búsqueda en un diccionario `pipe_grids_cat`.
```

```
# Definimos sus hiperparametros
reg_log_param = {"Modelo__penalty": [None,"l2"],
                 "Modelo__C": np.logspace(0, 4, 10),
                 "Modelo__class_weight": [None,"balanced"]}

rand_forest_param = {
    'Modelo__n_estimators': [10, 100, 200, 400],
    'Modelo__max_depth': [None,1,2,4,8],
    'Modelo__max_features': ['sqrt', 1, 2, 3],
    'Modelo__class_weight': [None,'balanced']}

param_grid_lgbm = {
    'Modelo__num_leaves': [15, 31, 50],
    'Modelo__learning_rate': [0.01, 0.05, 0.1],
    'Modelo__n_estimators': [50, 100, 200],
    'Modelo__max_depth': [-1,5, 10, 15],
    'Modelo__class_weight': [None,'balanced']}
```

```
cv = 5

gs_reg_log = GridSearchCV(logistic_pipeline,
                          reg_log_param,
                          cv=cv,
                          scoring="recall",
                          verbose=1,
                          n_jobs=-1)

gs_rand_forest = GridSearchCV(random_pipeline,
                          rand_forest_param,
                          cv=cv,
                          scoring="recall",
                          verbose=1,
                          n_jobs=-1)

gs_lgb = GridSearchCV(LGBM_pipeline,
                      param_gri
                      cv=cv,
                      scori
                      verb
                      n_j

pipe_grids_cat = {"gs_reg_l
        "gs_rand_forest":g
        "gs_lgb":gs_lgb}
```



Modelo preselección: LogisticRegression

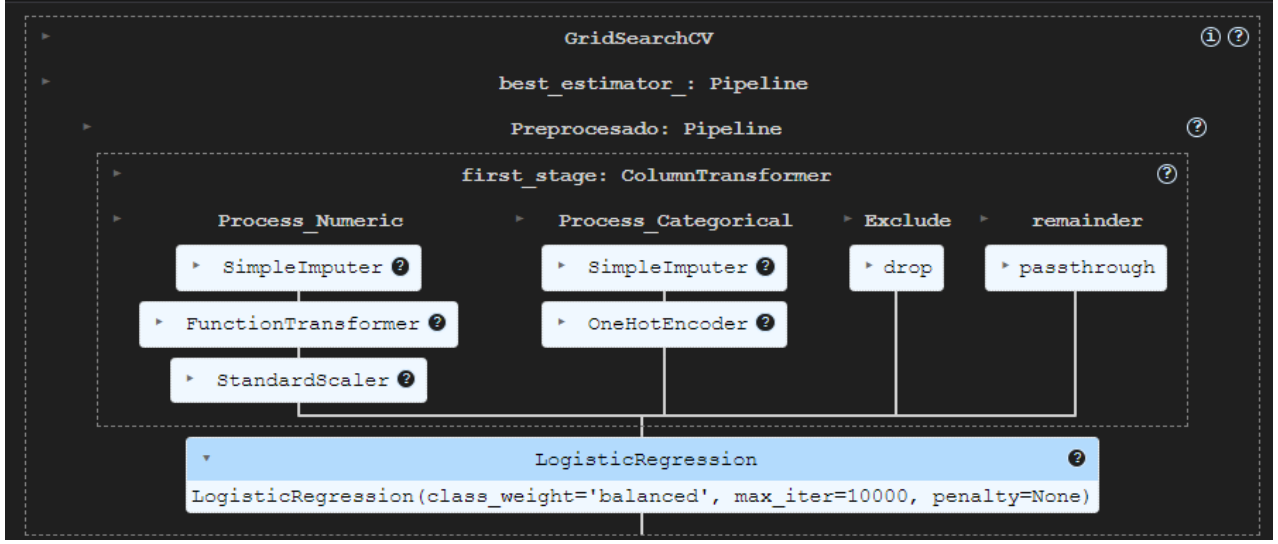
```
best_grids_cat  # Muestra los resultados
```

✓ 0.0s

	Grid	Best score
0	gs_reg_log	0.650000
1	gs_rand_forest	0.650000
2	gs_lgb	0.530882

```
# Selecting the best classification model
#
# - The best-performing model based on the GridSearchCV evaluation is selected.
# - The first row of the `best_grids_cat` DataFrame, which is already sorted by performance, is used.
# - This model will be stored in the `best_model_cat` variable for later use or saving.
best_model_cat = pipe_grids_cat[best_grids_cat.iloc[0, 0]]  # Extrae el mejor modelo
best_model_cat  # Muestra el modelo seleccionado
```

✓ 0.4s




```
# Guardado del mejor modelo de clasificación en formato pickle
#
# - Se asegura de que el directorio `src/models` exista para almacenar el modelo.
# - Se guarda el modelo en un archivo .pkl para su uso posterior.
# - El modelo guardado podrá ser cargado y utilizado sin necesidad de volver a entrenarlo.

# Asegurar que el directorio de modelos exista
os.makedirs('../models', exist_ok=True)

# Guardar el mejor modelo en formato pickle
with open('../models/modelo_pipeline_cat.pkl', 'wb') as archivo:
    pickle.dump(best_model_cat, archivo)  # Guarda el modelo en el archivo
```

```
# Recuperamos el modelo de pipelines (version pickle)
with open('../models/modelo_pipeline_cat.pkl', 'rb') as archivo: # ojo read binario
    modelo_pipeline_clf = pickle.load(archivo)
```

✓ 0.0s

```
print(classification_report(y_test_cat, modelo_pipeline_clf.predict(test)))
```

✓ 0.0s

```
              precision    recall  f1-score   support

           0       0.94      0.57      0.71       291
           1       0.09      0.57      0.16        23

    accuracy                           0.57       314
   macro avg       0.52      0.57      0.43       314
weighted avg       0.88      0.57      0.67       314
```

Conclusiones y acciones de mejora

8. Conclusions and actions for improvement

We have managed to provide the Business department, as requested, with a list of 106 entities that match the profile of our client interested in environmental issues and with areas for improvement. It is a more manageable list compared to the 1,573 entities we initially had.

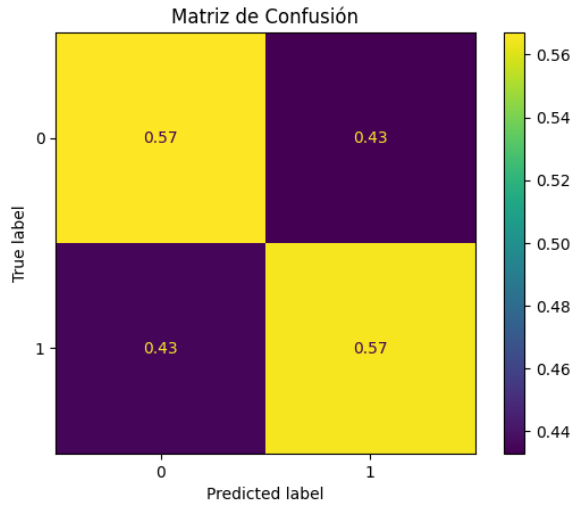
The final results of the marketing management will be interesting to validate our clustering and take adjustment measures.

Improvement actions: It is recommended to suggest to IAF to unify the types of surveys, restricting fields such as activity, year of creation, and number of employees, as their management becomes complex due to the wide margin left for natural language.

On the other hand, we have created a classification model for other companies different from the 1,573 with the RSA seal, leveraging the clustering. It is an additional tool to focus efforts on one type of entity or another. Thus, we can easily classify entities using 4 variables: organization type, activity classification, number of employees, and year of creation.

The chosen metric was recall since we aim to capture all relevant customers with a potential purchase intention, making it more important to minimize false negatives. This means we are willing to accept some false positives. The model was trained, achieving a recall of 65% in our target class during training and 57% during testing. It is not a highly accurate model due to limited data, but we consider it a reasonable initial model, as it does not indicate overfitting. The model detected 65 out of 100 favorable real cases in training and 57 out of 100 in testing.

Improvement actions: It is recommended to feed the model with more data to improve the metric. Furthermore, it remains a model trained with 1,573 data points within a specific type of entities interested in an RSA seal. It is possible that other types of entities may not show similar purchase behavior.



Gracias ;)

HTTPS://GITHUB.COM/
JL-PADILLA