

```

1  # -*- coding: utf-8 -*-
2
3  ### SCRIPT 20 - PYTHON
4  # GENERACION LIBRERIA CASOS SIMILARES
5  # PARA EJECUTAR EN GPT SIN ACCESO A PYTHON
6  # =====
7
8
9
10 # =====
11 # Generar biblioteca de casos en Markdown para modo "light" sin Python
12 # =====
13
14 import pandas as pd
15 import numpy as np
16 import json
17 from pathlib import Path
18
19 # -----
20 # 1. Rutas y configuración básica
21 # -----
22
23 excel_path =
24 "/content/drive/MyDrive/_Pipeline_produccion_prediccion/df_test_train_v8.xlsx"
25 output_md_path =
26 "/content/drive/MyDrive/_Pipeline_produccion_prediccion/casos_referencia_light.md"
27
28 # Columna de ID "de negocio" (si no existe se usará el índice del DataFrame)
29 id_col = "Identificador" # si no existe, se usará el índice
30
31 # Columna con el cluster final (objetivo usado para entrenar RF)
32 cluster_col = "Cluster_6"
33
34 # Columna con el ahorro final post-licitación
35 ahorro_col = "Ahorro_final"
36
37 # Variables PRE-licitación según tu tabla (numéricas y categóricas)
38 pre_numeric_cols = [
39     "Presupuesto_licitacion_lote_c", # presupuesto base de licitación (€)
40     "C_precio_p", # ponderación criterio precio (%)
41     "Plazo_m", # plazo del contrato (meses)
42     "N_lotes", # n° de lotes
43     "N_CPV", # n° de CPVs
44     "N_clasi_empresa", # n° clasificaciones exigidas
45     "C_juicio_valor_p_c", # ponderación juicio de valor (%)
46     "Intervalo_lici_d_c", # periodo de presentación ofertas (días)
47 ]
48
49 pre_categ_cols = [
50     "Tipo_de_contrato_c",
51     "Tipo_de_Administracion_c",
52     "Tipo_de_procedimiento_c",
53     "Tramitacion_c",
54     "Codigo_Postal_c",
55     "Tipo_ganador_lote_c",
56 ]
57
58 # Variables económico-competitivas principales para similitud
59 sim_vars = [
60     "Presupuesto_licitacion_lote_c",
61     "N_ofertantes",
62     "Plazo_m",
63     "Baja_p",
64     "C_precio_p",
65 ]
66
67 # Parámetro de saturación para similitud (%)
68 D_MAX = 5.0 # distancias >= D_MAX → similitud ≈ 0
69
70 # -----
71 # 2. Cargar DataFrame y verificar columnas

```

```

71 # -----
72
73 df = pd.read_excel(excel_path)
74
75 # Comprobaciones básicas
76 for col in [cluster_col, ahorro_col]:
77     if col not in df.columns:
78         raise ValueError(
79             f"La columna '{col}' no se encuentra en df_test_train_v8.xlsx. "
80             f"Ajusta 'cluster_col' / 'ahorro_col' en la cabecera del script."
81         )
82
83 # Avisos y limpieza de listas PRE
84 for col in pre_numeric_cols + pre_categ_cols:
85     if col not in df.columns:
86         print(f"AVISO: la columna PRE '{col}' no está en el DataFrame y se omitirá.")
87 pre_numeric_cols = [c for c in pre_numeric_cols if c in df.columns]
88 pre_categ_cols = [c for c in pre_categ_cols if c in df.columns]
89
90 # Avisos y limpieza de variables de similitud
91 for col in sim_vars:
92     if col not in df.columns:
93         print(f"AVISO: la columna de similitud '{col}' no está en el DataFrame y se
94             omitirá.")
95 sim_vars = [c for c in sim_vars if c in df.columns]
96
97 if not sim_vars:
98     raise ValueError("No queda ninguna variable en 'sim_vars'; revisa que existan en
99         el Excel.")
100
101 # -----
102 # 3. Cálculo de medias, desviaciones y z-scores
103 # -----
104
105 stats = {}
106 for col in sim_vars:
107     stats[col] = {
108         "mean": float(df[col].mean()),
109         "std": float(df[col].std(ddof=0)), # desviación poblacional
110     }
111
112 for col in sim_vars:
113     mean = stats[col]["mean"]
114     std = stats[col]["std"]
115     if std == 0 or np.isnan(std):
116         df[f"z_{col}"] = np.nan
117         print(f"AVISO: desviación nula/NaN en {col}; z_{col} se rellena con NaN.")
118     else:
119         df[f"z_{col}"] = (df[col] - mean) / std
120
121 def fmt_float(x):
122     """Formateo seguro para floats/ints → float con 6 decimales, o None si NaN."""
123     if pd.isna(x):
124         return None
125     return round(float(x), 6)
126
127 # -----
128 # 4. Cabecera del .md con instrucciones para el GPT
129 # -----
130
131 header_lines = []
132
133 header_lines.append("# Biblioteca de casos de referencia - Riesgo en licitaciones
134     (modo light sin Python)\n")
135 header_lines.append("## Instrucciones para el GPT\n")
136 header_lines.append(
137     "Este archivo contiene una colección de casos históricos (TRAIN+TEST) del DSS de
138     riesgo en "
139     "licitaciones públicas. Cada caso está definido en un bloque `json`
140     independiente.\n"

```

```

138 )
139
140 # Variables PRE
141 header_lines.append("### Variables de PRE-licitación a utilizar en la búsqueda\n")
142 header_lines.append(
143     "Debes buscar similitudes **solo** sobre variables de PRE-licitación. Para cada
    caso, las "
144     "variables PRE disponibles (si existen en el dataset) son:\n"
145 )
146 if pre_numeric_cols:
147     header_lines.append(f"- Numéricas: {' ', '.join(pre_numeric_cols)}")
148 if pre_categ_cols:
149     header_lines.append(f"- Categóricas: {' ', '.join(pre_categ_cols)}")
150 header_lines.append("")
151
152 header_lines.append(
153     "En la búsqueda semántica, da más peso a las siguientes variables numéricas de
    PRE, que son "
154     "especialmente representativas:\n"
155 )
156 header_lines.append("- `Presupuesto_licitacion_lote_c` (presupuesto base de
    licitación del lote, €)")
157 header_lines.append("- `C_precio_p` (ponderación del precio en el criterio de
    adjudicación, %)")
158 header_lines.append("- `Plazo_m` (plazo contractual en meses)\n")
159
160 # Variables económicas principales
161 header_lines.append("### Variables económico-competitivas principales\n")
162 header_lines.append(
163     "Cada caso incluye, además, las cinco variables económico-competitivas
    principales, que usarás "
164     "tanto para la explicación como para el cálculo de similitud:\n"
165 )
166 for col in sim_vars:
167     header_lines.append(f"- `{col}`")
168 header_lines.append("- `Ahorro_final` (resultado post-licitación)\n")
169
170 # Medias y desviaciones
171 header_lines.append("### Coordenadas normalizadas para similitud\n")
172 header_lines.append(
173     "Para los casos que recuperes del índice (normalmente los **3 más parecidos** que
    el RAG te "
174     "devuelva), utiliza las coordenadas normalizadas (`z_*`) de estas variables. Se
    han calculado "
175     "con media 0 y desviación típica 1 sobre toda la base histórica.\n"
176 )
177 header_lines.append("Parámetros usados (media y desviación) por variable:\n")
178
179 for col in sim_vars:
180     header_lines.append(
181         f"- {col}: mean = {stats[col]['mean']:.6f}, std = {stats[col]['std']:.6f}"
182     )
183
184 header_lines.append("")
185 header_lines.append("### Cálculo del % de similitud entre un caso nuevo y un caso de
    referencia\n")
186 header_lines.append("1. A partir de las respuestas del usuario, estima los valores
    de:")
187 header_lines.append(f"    {' ', '.join(sim_vars)}")
188 header_lines.append(
189     "2. Calcula sus z-scores usando las medias y desviaciones anteriores:\n"
190     "    `z_x = (x - mean_x) / std_x`.\n"
191 )
192 header_lines.append(
193     "3. Para cada caso de referencia recuperado del índice, usa los valores "
194     "`z_...` contenidos en el bloque JSON y computa la distancia euclídea:\n"
195 )
196 header_lines.append("    `d = sqrt( sum_k (z_k_nuevo - z_k_referencia)^2 )`\n")
197 header_lines.append(
198     f"4. Convierte esta distancia en un porcentaje de similitud:\n\n"
199     f"    `similitud = max(0, 100 * (1 - d / {D_MAX:.1f}))`\n"
200     f"    donde `D_MAX = {D_MAX:.1f}` es un valor de saturación: distancias `d ≥"

```

```

201     D_MAX` se consideran "
202     "similitud  $\approx 0$ .\n"
203 )
204 header_lines.append(
205     "Utiliza esta similitud solo para comparar **entre** los pocos casos devueltos
206     por el RAG "
207     "(por ejemplo, los 3 más cercanos) y selecciona el caso con mayor similitud como
208     el más "
209     "representativo."
210 )
211 header_lines.append(
212     "Si los 3 casos pertenecen a clusters diferentes, adopta como *cluster de
213     referencia* el "
214     "cluster del caso con mayor similitud.\n"
215 )
216 header_lines.append("---\n")
217 header_lines.append("## Casos de referencia\n")
218 header_lines.append(
219     "A continuación se listan todos los casos de la base de datos (TRAIN+TEST). "
220     "Cada bloque `json` describe un expediente:\n"
221 )
222 header_text = "\n".join(header_lines)
223
224 # -----
225 # 5. Construcción de los bloques JSON por caso
226 # -----
227
228 lines = [header_text]
229
230 for idx, row in df.iterrows():
231     # case_id: preferentemente Identificador si existe, si no, índice del DataFrame
232     if id_col in df.columns:
233         raw_id = row[id_col]
234     else:
235         raw_id = idx
236     case_id = raw_id if not pd.isna(raw_id) else idx
237
238     block = {
239         "case_id": case_id,
240         "dataset_index": int(idx),
241         "cluster_final": row.get(cluster_col, None),
242         "Ahorro_final": fmt_float(row.get(ahorro_col, np.nan)),
243     }
244
245     # Variables económico-competitivas + sus z-scores (si existen)
246     for col in sim_vars:
247         block[col] = fmt_float(row.get(col, np.nan))
248         zcol = f"z_{col}"
249         if zcol in df.columns:
250             block[zcol] = fmt_float(row.get(zcol, np.nan))
251
252     # Variables PRE-licitación agrupadas en 'pre_variables'
253     pre_vars = {}
254     for col in pre_numeric_cols + pre_categ_cols:
255         val = row.get(col, np.nan)
256         if isinstance(val, (np.floating, float, int, np.integer)):
257             val_out = fmt_float(val)
258         else:
259             val_out = None if pd.isna(val) else val
260         pre_vars[col] = val_out
261     block["pre_variables"] = pre_vars
262
263     # Serialización a JSON embebido en Markdown
264     block_json = json.dumps(block, ensure_ascii=False, indent=2)
265     lines.append(f"### CASE_ID = {case_id}\n\n```\njson\n{block_json}\n```\n")
266
267 md_text = "\n".join(lines)
268 Path(output_md_path).write_text(md_text, encoding="utf-8")

```

```
269     print(f"Archivo Markdown generado en:\n{output_md_path}")
```