Examen Módulo 1 — Ciencia de Datos (G33)

Alumno: David Magaña Celis

Objetivo

Desarrollar un flujo reproducible para **cargar, validar, perfilar y transformar** los datos, definir la **unidad muestral**, construir una base analítica y aplicar técnicas de **limpieza, reducción y selección de variables**.

Entregables

- Notebook: `reporte.ipynb`
- Imagen: `resultado/pca_2d.png`
- Tablas exportadas: `resultado/usuarios.csv`, `resultado/restaurantes.csv`

Librerías

In [13]:
```
import pandas as pd
import examen
import examen.cargar_datos as cd
import examen.validar_datos as vd
import examen.perfil_datos as ped
import examen.analiticas as an
import examen.modelado as mo
from IPython.display import Image, display
from importlib import reload

pd.set_option("display.max_rows", 120)
pd.set_option("display.min_rows", 20)
pd.set_option("display.max_columns", 50)
pd.set_option("display.max_colwidth", 80)
pd.set_option("display.width", 120)
pd.set_option("display.expand_frame_repr", True)

cd, vd, ped, an, mo = map(reload, (cd, vd, ped, an, mo))
```

Extracción

Se crearon dos módulos dentro del paquete `examen`:

- `cargar_datos.py`: carga de tablas desde el archivo .zip ubicado en `data/` (sin extraer a disco).
- `validar_datos.py`: validación simple comparando pares CSV vs XLSX.

Tras generar reportes completos de diferencias (NaN-safe) para los pares disponibles, se decidió utilizar los archivos CSV como fuente principal para uniones (joins) y pasos posteriores, manteniendo los XLSX como referencia.

In [14]:
```
dfs = cd.cargar_dfs_desde_zip_en_data()
list(dfs.keys())
```

```
Leyendo datos/parking.csv -> key=parking_csv
Leyendo datos/usercuisine.csv -> key=usercuisine_csv
Leyendo datos/restaurants.xlsx -> key=restaurants_xlsx
Leyendo datos/users.xlsx -> key=users_xlsx
Leyendo datos/restaurants.csv -> key=restaurants_csv
Leyendo datos/cuisine.csv -> key=cuisine_csv
Leyendo datos/users.csv -> key=users_csv
Leyendo datos/ratings.csv -> key=ratings_csv
Leyendo datos/hours.csv -> key=hours_csv
Leyendo datos/userpayment.csv -> key=userpayment_csv
Leyendo datos/payment_methods.csv -> key=payment_methods_csv
```

Out[14]:
```
['parking_csv',
 'usercuisine_csv',
 'restaurants_xlsx',
 'users_xlsx',
 'restaurants_csv',
 'cuisine_csv',
 'users_csv',
 'ratings_csv',
 'hours_csv',
 'userpayment_csv',
 'payment_methods_csv']
```

In [15]:
```
reps = vd.reporte_todos_pares_completo(dfs)

for base, rep in reps.items():
    print(rep["summary"])
    display(rep["diffs"])
```

```
{'base': 'restaurants', 'csv_shape': (130, 21), 'xlsx_shape': (130, 21), 'sa
me_columns': True, 'equals_pandas': False, 'mismatch_cells': 107, 'note':
''}
```

	row	col	csv	xlsx
0	1	zip	78280	78280
1	2	latitude	22.149709	22.149709
2	2	longitude	-100.976093	-100.976093
3	2	zip	78000	78000
4	3	city	victoria	victoria
5	5	zip	74000	74000
6	6	latitude	23.754357	23.754357
7	6	name	Taqueria EL amigo	Taqueria EL amigo
8	6	zip	87018	87018
9	8	latitude	23.760268	23.760268
10	8	longitude	-99.165865	-99.165865
11	9	zip	78421	78421
12	10	zip	78421	78421
13	11	zip	78399	78399
14	12	zip	78250	78250
15	13	zip	78395	78395
16	14	latitude	22.141421	22.141421
17	14	zip	78290	78290
18	17	name	rockabilly	rockabilly
19	20	zip	78399	78399
20	21	latitude	22.150643	22.150643
21	22	zip	62250	62250
22	25	zip	78269	78269
23	26	address	avenida salvador montiel	avenida salvador montiel
24	26	zip	62320	62320
25	28	zip	78000	78000
26	29	latitude	22.144337	22.144336
27	32	zip	62460	62460
28	33	zip	78430	78430
29	35	latitude	22.14788	22.14788
30	35	longitude	-100.989472	-100.989472
31	36	zip	78390	78390

	row	col	csv	xlsx
32	37	zip	78000	78000
33	42	longitude	-101.002958	-101.002958
34	42	zip	78200	78200
35	47	zip	78250	78250
36	48	zip	62170	62170
37	49	zip	78000	78000
38	50	latitude	22.145008	22.145008
39	50	zip	78270	78270
40	51	zip	62790	62790
41	52	zip	78038	78038
42	53	latitude	22.140517	22.140517
43	53	zip	78210	78210
44	54	latitude	22.156883	22.156883
45	54	longitude	-100.978485	-100.978485
46	54	zip	78310	78310
47	55	zip	78270	78270
48	56	latitude	22.136253	22.136253
49	57	zip	78000	78000
50	59	zip	78740	78740
51	60	zip	78000	78000
52	61	zip	78000	78000
53	62	longitude	-101.019845	-101.019845
54	63	zip	78000	78000
55	64	zip	62290	62290
56	65	latitude	22.150981	22.150981
57	65	zip	78000	78000
58	69	zip	78250	78250
59	70	latitude	22.139578	22.139578
60	72	latitude	23.752982	23.752982
61	72	longitude	-99.168434	-99.168434
62	73	latitude	23.752931	23.75293
63	74	zip	78310	78310

	row	col	csv	xlsx
64	75	longitude	-100.985809	-100.985809
65	76	latitude	23.736798	23.736798
66	77	zip	78240	78240
67	78	longitude	-101.018032	-101.018032
68	78	zip	78210	78210
69	79	zip	78396	78396
70	80	zip	78349	78349
71	81	latitude	22.142273	22.142273
72	81	name	el lechon potosino	el lechon potosino
73	81	city	san luis potosi	san luis potosi
74	82	latitude	22.142017	22.142017
75	82	zip	78269	78269
76	85	zip	78433	78433
77	88	latitude	18.869993	18.869993
78	90	latitude	22.173596	22.173595
79	91	zip	62290	62290
80	92	latitude	22.141647	22.141647
81	96	latitude	23.732423	23.732423
82	97	longitude	-100.974494	-100.974494
83	99	zip	79300	79300
84	101	latitude	22.153324	22.153324
85	103	latitude	23.735523	23.735523
86	103	city	victoria	victoria
87	104	zip	78214	78214
88	105	longitude	-100.974269	-100.974269
89	105	zip	78000	78000
90	106	latitude	23.752304	23.752304
91	106	longitude	-99.166913	-99.166913
92	108	zip	78269	78269
93	109	latitude	23.752943	23.752943
94	111	longitude	-99.165709	-99.165709
95	113	zip	78000	78000

	row	col	csv	xlsx
96	116	zip	64000	64000
97	119	latitude	22.152481	22.152481
98	119	zip	78000	78000
99	121	zip	78000	78000
100	124	latitude	18.92229	18.92229
101	124	zip	62000	62000
102	126	latitude	22.149192	22.149192
103	126	zip	78220	78220
104	128	latitude	18.875011	18.875011
105	128	longitude	-99.159422	-99.159422
106	129	latitude	22.135364	22.135364


```
{'base': 'users', 'csv_shape': (138, 19), 'xlsx_shape': (138, 19), 'same_col
umns': True, 'equals_pandas': False, 'mismatch_cells': 173, 'note': ''}
```

	row	col	csv	xlsx
0	0	smoker	false	False
1	1	smoker	false	False
2	2	smoker	false	False
3	3	longitude	-99.183	-99.183
4	3	smoker	false	False
5	4	smoker	false	False
6	5	longitude	-100.983	-100.983
7	5	smoker	true	True
8	6	latitude	22.118464	22.118464
9	6	smoker	false	False
...	...	...	...	...
163	131	longitude	-100.983	-100.983
164	131	smoker	false	False
165	132	latitude	18.886698	18.886698
166	132	smoker	false	False
167	133	smoker	false	False
168	134	smoker	false	False
169	135	smoker	true	True
170	136	longitude	-100.944623	-100.944623
171	136	smoker	false	False
172	137	smoker	false	False

173 rows × 4 columns

In [16]:
```
dfs_csv = cd.obtener_csvs(dfs)
```

Perfilado de DataFrames (CSV)

En esta sección se realiza el **perfilado exploratorio** de todos los DataFrames provenientes de archivos **CSV**, con el objetivo de:

- identificar estructura (filas/columnas),
- revisar distribución y tipo de variables,
- detectar valores faltantes y posibles inconsistencias,
- y documentar la calidad inicial de los datos antes de la integración/modelado.

In [17]:

```
dfs_perfil = ped.perfilar_todos_los_dfs(dfs_csv)

partes = []
for name, p in dfs_perfil.items():
    if p is None or p.empty:
        continue

    p = p.dropna(axis=1, how="all")
    partes.append(p.assign(df=name))

perfil_all = pd.concat(partes, ignore_index=True) if partes else pd.DataFram

cols = ["df","Variable","Tipo_inferido","Dtype","Nulos","%Nulos","Unicos","%
perfil_all = perfil_all[[c for c in cols if c in perfil_all.columns]]

perfil_all
```

Out[17]:

	df	Variable	Tipo_inferido	Dtype	Nulos	%Nulos	Un
0	users_csv	userID	Texto	object	0	0.0	
1	users_csv	latitude	Numerica	float64	0	0.0	
2	users_csv	longitude	Numerica	float64	0	0.0	
3	users_csv	weight	Numerica	int64	0	0.0	
4	users_csv	height	Numerica	float64	0	0.0	
5	users_csv	birth_year	Numerica	int64	0	0.0	
6	users_csv	color	Categorica	object	0	0.0	
7	users_csv	dress_preference	Categorica	object	0	0.0	
8	users_csv	interest	Categorica	object	0	0.0	
9	users_csv	religion	Categorica	object	0	0.0	
10	users_csv	activity	Categorica	object	0	0.0	
11	users_csv	ambience	Categorica	object	0	0.0	
12	users_csv	transport	Categorica	object	0	0.0	
13	users_csv	marital_status	Categorica	object	0	0.0	
14	users_csv	hijos	Categorica	object	0	0.0	
15	users_csv	personality	Categorica	object	0	0.0	
16	users_csv	budget	Categorica	object	0	0.0	
17	users_csv	smoker	Categorica	object	0	0.0	
18	users_csv	drink_level	Categorica	object	0	0.0	
19	usercuisine_csv	userID	Texto	object	0	0.0	
20	usercuisine_csv	Rcuisine	Texto	object	0	0.0	
21	userpayment_csv	userID	Texto	object	0	0.0	
22	userpayment_csv	Upayment	Categorica	object	0	0.0	
23	ratings_csv	userID	Categorica	object	0	0.0	
24	ratings_csv	placeID	Numerica	int64	0	0.0	
25	ratings_csv	rating	Numerica	int64	0	0.0	

	df	Variable	Tipo_inferido	Dtype	Nulos	%Nulos	Un
26	ratings_csv	food_rating	Numerica	int64	0	0.0	
27	ratings_csv	service_rating	Numerica	int64	0	0.0	
28	restaurants_csv	placeID	Numerica	int64	0	0.0	
29	restaurants_csv	the_geom_meter	Texto	object	0	0.0	
30	restaurants_csv	latitude	Numerica	float64	0	0.0	
31	restaurants_csv	longitude	Numerica	float64	0	0.0	
32	restaurants_csv	name	Texto	object	0	0.0	
33	restaurants_csv	address	Texto	object	0	0.0	
34	restaurants_csv	zip	Texto	object	0	0.0	
35	restaurants_csv	city	Categorica	object	0	0.0	
36	restaurants_csv	url	Categorica	object	0	0.0	
37	restaurants_csv	state	Categorica	object	0	0.0	
38	restaurants_csv	smoking_area	Categorica	object	0	0.0	
39	restaurants_csv	country	Categorica	object	0	0.0	
40	restaurants_csv	alcohol	Categorica	object	0	0.0	
41	restaurants_csv	dress_code	Categorica	object	0	0.0	
42	restaurants_csv	accessibility	Categorica	object	0	0.0	
43	restaurants_csv	price	Categorica	object	0	0.0	
44	restaurants_csv	other_services	Categorica	object	0	0.0	
45	restaurants_csv	Rambience	Categorica	object	0	0.0	
46	restaurants_csv	franchise	Categorica	object	0	0.0	
47	restaurants_csv	area	Categorica	object	0	0.0	
48	restaurants_csv	fax	Categorica	object	0	0.0	
49	parking_csv	placeID	Numerica	int64	0	0.0	
50	parking_csv	parking_lot	Categorica	object	0	0.0	
51	cuisine_csv	placeID	Numerica	int64	0	0.0	

	df	Variable	Tipo_inferido	Dtype	Nulos	%Nulos	Un
52	cuisine_csv	Rcuisine	Categorica	object	0	0.0	
53	payment_methods_csv	placeID	Numerica	int64	0	0.0	
54	payment_methods_csv	Rpayment	Categorica	object	0	0.0	
55	hours_csv	placeID	Numerica	int64	0	0.0	
56	hours_csv	hours	Categorica	object	0	0.0	
57	hours_csv	days	Categorica	object	0	0.0	

In [18]:
```
ped.reporte_relacional(dfs_csv, ratings_key="ratings_csv")
```

Out[18]:

	df	key	rows	unique_keys	min_per_key	max_per_key	me
0	users_csv	userID	138	138	1	1	
1	usercuisine_csv	userID	330	138	1	103	
2	userpayment_csv	userID	177	133	1	4	
3	ratings_csv	userID	1161	138	3	18	
4	ratings_csv	placeID	1161	130	3	36	
5	restaurants_csv	placeID	130	130	1	1	
6	parking_csv	placeID	702	675	1	3	
7	cuisine_csv	placeID	916	769	1	9	
8	payment_methods_csv	placeID	1314	615	1	8	
9	hours_csv	placeID	2339	694	2	12	

Unidad muestral y modelo relacional

A partir del **reporte relacional** (`reporte_relacional`) se identificó que la tabla `ratings_csv` funciona como **tabla de hechos**, ya que contiene directamente las variables de calificación (`rating` , `food_rating` , `service_rating`) y conecta a usuarios y restaurantes mediante las llaves `userID` y `placeID` .

Evidencia y justificación

- **Tabla de hechos:** `ratings_csv` (1161 filas).
- **Llaves presentes:** `userID` (138 únicos) y `placeID` (130 únicos).
- **Unidad muestral definida: 1 fila = 1 evaluación** de un usuario (`userID`) a un restaurante (`placeID`).

- **Llave lógica de la evaluación:** (`userID` , `placeID`), y se verificó que **no existen duplicados** para este par, por lo que cada registro representa una evaluación única.

En consecuencia, la base principal para análisis y modelado se construye **partiendo de** `ratings_csv` , preservando su número de filas y, por lo tanto, su unidad muestral.

Dimensiones analíticas para enriquecer la base

Con el mismo reporte relacional se observó que:

- `users_csv` es **1 a 1** con `userID` (mean_per_key = 1), por lo que puede integrarse directamente a nivel usuario.
- `restaurants_csv` es **1 a 1** con `placeID` (mean_per_key = 1), por lo que puede integrarse directamente a nivel restaurante.
- Otras tablas (`usercuisine_csv` , `userpayment_csv` , `cuisine_csv` , `payment_methods_csv` , `hours_csv` , `parking_csv`) presentan relaciones **1 a muchos** respecto a `userID` o `placeID` , por lo que se transforman previamente mediante **agregaciones** (conteos y listas de categorías) para mantener una sola fila por llave.

Para mantener el notebook legible y robustecer la paquetería, se implementaron funciones que construyen:

- `dim_user` (**usuarios.csv**): tabla analítica a nivel `userID` (1 fila por usuario), integrando `users_csv` y agregados por `userID` .
- `dim_place` (**restaurantes.csv**): tabla analítica a nivel `placeID` (1 fila por restaurante), integrando `restaurants_csv` y agregados por `placeID` .

Finalmente, la **base de modelado** se obtiene realizando `LEFT JOIN` desde `ratings_csv` hacia `dim_user` y `dim_place` , asegurando que el número de filas permanezca en 1161 y que la unidad muestral continúe siendo **la evaluación usuario–restaurante**.

In [19]:
```
dim_user = an.construir_dim_usuario(dfs_csv)
dim_place = an.construir_dim_restaurante(dfs_csv)
an.exportar_tablas_analiticas(dim_user, dim_place, out_dir="./resultado")

base = an.construir_base_modelado(dfs_csv, dim_user, dim_place)
```

Ingeniería de datos: variables nuevas a nivel de la unidad muestral (`ratings`)

Con la unidad muestral definida como **1 fila = 1 evaluación** (`userID` , `placeID`), se construyeron variables adicionales **derivadas directamente de las calificaciones** de

`ratings_csv`. Estas variables se calculan al mismo grano de la unidad muestral (no generan explosión de filas) y buscan capturar:

- La **intensidad promedio** de la evaluación (señal global).
- La **consistencia o discrepancia** entre componentes (comida vs servicio).
- El **contexto del evaluador** (tendencia típica del usuario) y del evaluado (tendencia típica del restaurante), calculados con enfoque *leave-one-out* para no usar la misma observación en su propio promedio.

Variables creadas (≥ 5) y justificación

- **`subrating_mean`**: resume en una sola señal el nivel promedio entre `food_rating` y `service_rating`, útil para modelos que aprovechan una medida global de calidad.
- **`subrating_gap`**: captura si la evaluación favorece más la comida o el servicio; ayuda a distinguir perfiles donde una dimensión domina a la otra.
- **`subrating_gap_abs`**: mide la consistencia entre comida y servicio; discrepancias grandes pueden indicar experiencias "mixtas" que un promedio no detecta.
- **`user_mean_rating_loo`**: representa la tendencia típica del usuario a calificar alto/bajo (sesgo del evaluador) sin incluir la evaluación actual; permite ajustar por usuarios más exigentes o más complacientes.
- **`place_mean_rating_loo`**: representa la tendencia típica del restaurante a recibir mejores/peores calificaciones sin incluir la evaluación actual; provee un contexto de reputación promedio a nivel `placeID`.

In [20]:
```
base = an.agregar_features_ratings(base)

cols_new = [
    "subrating_mean",
    "subrating_gap",
    "subrating_gap_abs",
    "user_mean_rating_loo",
    "place_mean_rating_loo",
]

display(base[cols_new].head() )

display(base.shape[1])
```

	subrating_mean	subrating_gap	subrating_gap_abs	user_mean_rating_loo	place_m
0	2.0	0	0	1.250000	
1	1.5	1	1	1.250000	
2	2.0	0	0	1.250000	
3	2.0	0	0	1.500000	
4	1.5	-1	1	0.571429	

66

Construcción de variable objetivo

Se definió una **variable objetivo categórica binaria** alineada a la unidad muestral (1 fila = 1 evaluación).
El objetivo consiste en identificar evaluaciones **altas** del restaurante.

- **Objetivo (`target`)**: 1 si el usuario otorgó la calificación máxima (**rating = 2**), 0 en caso contrario.
- Este planteamiento permite:
  - formular un problema de **clasificación** (alto vs no-alto),
  - habilitar técnicas como **SelectKBest** y **WoE/IV** (que requieren objetivo),
  - mantener coherencia con el grano del dataset (no se agregan filas).

Se conserva la columna `rating` como variable original y se agrega `target` como columna adicional.

In [21]:
```
base_t = mo.construir_objetivo(
    base,
    tipo="binaria",
    regla_binaria="rating_eq_2",
    col_rating="rating",
    out_col="target",
)

base_t["target"].value_counts(dropna=False), base_t["target"].value_counts(r
```

Out[21]:
```
(target
 0    675
 1    486
 Name: count, dtype: Int64,
 target
 0    0.5814
 1    0.4186
 Name: proportion, dtype: Float64)
```

Limpieza de datos

La limpieza se implementa con un enfoque reproducible: primero se genera evidencia (tablas) y luego se aplican reglas.

Procesos evaluados:

- **Valores extremos**: se revisan únicamente variables continuas plausibles (p.ej. edad/peso/altura/distancia).
  Las variables `rating`, `food_rating` y `service_rating` no se tratan como outliers porque están acotadas (0–2).
- **Variables poco pobladas (65%)**: se identifican columnas con porcentaje de nulos superior al umbral.
- **Valores ausentes**: se cuantifican por columna y se tratan mediante imputación en preprocesamiento (mediana/moda).
- **Correlación alta**: se remueven **solo** variables con correlación perfecta **|corr| = 1**
- **Variables unarias**: se eliminan columnas con 0 o 1 valor distinto (incluyendo NA), ya que no aportan señal.

Cuando algún proceso no aplica (por ejemplo, no hay pares con |corr|=1), se justifica mostrando la evidencia.

In [22]:
```
y = base_t["target"].copy()
X = base_t.drop(columns=["target"])

rep = mo.reporte_limpieza(X, missing_threshold=0.65)

print("=== Variables poco pobladas (flag 65%) ===")
display(rep["missing"].query("flag_poco_poblada == True").head(50))

print("=== Variables unarias ===")
display(rep["unarias"])

print("=== Pares con |corr| = 1 (si existen) ===")
display(rep["corr_abs_eq_1"])
```

```
=== Variables poco pobladas (flag 65%) ===
```

	col	pct_null	flag_poco_poblada
0	fax	100.00	True
1	url	89.41	True

```
=== Variables unarias ===
```

	col	nunique_incl_na
0	n_rest_parking_types	1
1	fax	1

```
=== Pares con |corr| = 1 (si existen) ===
```

	col_a	col_b	corr
0	birth_year	age_ref	-1.0

In [23]:
```
cfg = mo.LimpiezaConfig(
    missing_threshold=0.65,
    drop_missing=True,
    drop_unarias=True,
    drop_corr_abs_eq_1=True,
    outlier_strategy="winsorize",
)

X_clean, rep_apply = mo.aplicar_limpieza(X, config=cfg)
rep_apply
```

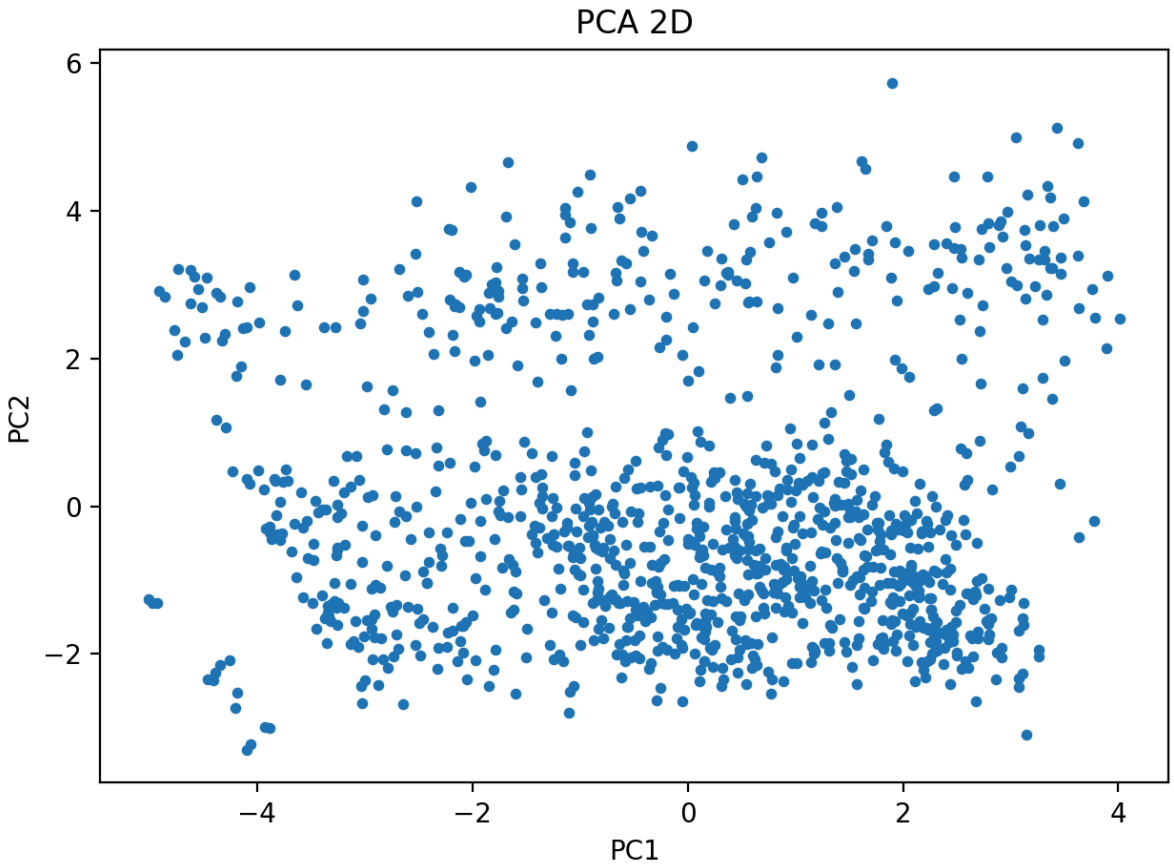
Out[23]:
```
{'shape_before': (1161, 66),
 'drop_poco_pobladas': ['fax', 'url'],
 'drop_unarias': ['n_rest_parking_types'],
 'drop_corr_abs_eq_1': ['age_ref'],
 'outlier_cols': ['weight', 'height'],
 'outlier_strategy': 'winsorize',
 'shape_after': (1161, 62)}
```

Reducción de dimensiones (no secuencial)

A partir del conjunto resultante de la limpieza (`X_clean`) se aplicaron técnicas **independientes** (no secuenciales):

1. **PCA a 2 dimensiones** para visualizar la estructura del dataset en un espacio reducido.
   Se genera un archivo **.png** como entregable.
2. **Clustering de variables** para mitigar multicolinealidad: se agrupan variables similares (por 1–|corr|) y se selecciona un representante por cluster (por mayor información mutua con el objetivo, cuando aplica).
3. **SelectKBest** para seleccionar variables con mayor asociación estadística con el objetivo.
4. **WoE e IV** (solo aplica por objetivo binario): se rankean variables por su poder de separación entre clases.

In [24]:
```
pca_out = mo.pca_2d_y_png(X_clean, out_png="./resultado/pca_2d.png")
pca_out["explained_var"], pca_out["png_path"]
display(Image(filename="resultado/pca_2d.png"))
```



In [25]:
```
pre = mo.construir_preprocesador(X_clean)
pre.fit(X_clean, y)

for name, trans, cols in pre.transformers_:
    print("\n=== Transformer:", name, "===")
    print("Type:", type(trans))
    if hasattr(trans, "steps"):  # Pipeline
        print("Pipeline steps:", [s[0] for s in trans.steps])
```

```
=== Transformer: num ===
Type: <class 'sklearn.pipeline.Pipeline'>
Pipeline steps: ['san', 'imp', 'sc']

=== Transformer: cat ===
Type: <class 'sklearn.pipeline.Pipeline'>
Pipeline steps: ['san', 'imp', 'oh']
```

In [26]:
```
rep_vars = mo.clustering_variables_representantes(
    X_clean,
    y=y,
    distance_threshold=0.30,
    top_n=25,
)
display(rep_vars)
```

	cluster	feat
0	15	num__ra
1	343	num__user_mean_rating
2	239	num__place_mean_rating
3	17	num__latit
4	90	num__longit
5	144	cat__hours_Mon_10:30-22
6	331	cat__interest_eco-frie
7	327	num__we
8	315	cat__hours_Tue_10:30-23
9	243	cat__userID_U1
10	10	num__plac
11	107	cat__hours_Sat_19:00-23
12	34	cat__address_Venustiano Carranza 2175 Ja
13	261	num__he
14	368	cat__marital_status_wi
15	185	cat__the_geom_meter_0101000020957F0000F9F19DDC3B4858C191B265A83BA44
16	316	cat__rest_cuisines_Bakery | Cafet
17	93	cat__zip_78
18	2	cat__user_cuisines_Latin_Amer
19	29	cat__hours_Fri_11:30-19
20	179	cat__hours_Sun_07:00-23
21	130	cat__hours_Sat_09:00-12
22	112	cat__hours_Fri_20:00-0
23	79	cat__the_geom_meter_0101000020957F0000245028A70E4758C18E383571B7A84
24	288	cat__userID_U1

In [27]:
```
top_kbest = mo.selectkbest_top(X_clean, y=y, k=20)
display(top_kbest)
```

	feature	f_score
0	num__rating	3915.643107
1	num__subrating_mean	778.510675
2	num__food_rating	582.594657
3	num__service_rating	564.027244
4	num__user_mean_rating_loo	297.536290
5	cat__interest_eco-friendly	34.661749
6	num__latitude	24.891829
7	cat__color_blue	24.711953
8	num__birth_year	23.506563
9	cat__drink_level_casual drinker	18.845537
10	num__place_mean_rating_loo	18.837187
11	cat__color_purple	18.107765
12	num__latitude_rest	16.952430
13	cat__price_low	16.902403
14	cat__transport_public	16.362605
15	cat__dress_code_formal	15.316443
16	cat__userID_U1137	15.316443
17	cat__drink_level_social drinker	15.110670
18	cat__color_white	15.055457
19	cat__dress_preference_informal	14.979008

In [28]:
```
iv_rank = mo.ranking_iv(X_clean, y_bin=y, n_bins=5)
display(iv_rank.head(30))
```

	variable	iv
0	rating	39.669721
1	userID	7.166492
2	subrating_mean	2.853139
3	the_geom_meter	2.375782
4	name	2.373427
5	food_rating	1.980128
6	service_rating	1.737799
7	address	1.712740
8	user_cuisines	1.351892
9	user_mean_rating_loo	1.196137
10	hours_Fri	0.908864
11	hours_Thu	0.908864
12	hours_Wed	0.908864
13	hours_Tue	0.908864
14	hours_Mon	0.908864
15	hours_Sun	0.798939
16	hours_Sat	0.786572
17	zip	0.736345
18	rest_cuisines	0.378494
19	activity	0.369744
20	color	0.222506
21	rest_payments	0.219103
22	user_payments	0.206911
23	interest	0.174991
24	place_mean_rating_loo	0.138976
25	latitude	0.130368
26	latitude_rest	0.107513
27	birth_year	0.096909
28	drink_level	0.081221
29	state	0.079220

Cuestionario

1) ¿Por qué Excel no es una Base de Datos?

Excel es una herramienta de hoja de cálculo, no un motor de base de datos. Aunque permite almacenar datos, no está diseñado para:

- garantizar **integridad** (restricciones, llaves foráneas, reglas consistentes),
- manejar **concurrencia** (múltiples usuarios modificando con control transaccional),
- asegurar **trazabilidad y auditoría** robusta,
- optimizar consultas complejas a gran escala (índices, planes de ejecución),
- aplicar transacciones ACID de forma nativa. Por ello, Excel sirve para análisis y reporteo, pero no sustituye un sistema de gestión de bases de datos (DBMS).

2) Diferencia entre Ingeniero de Datos, Científico de Datos y Arquitecto de Datos

- **Ingeniero de Datos**: construye y mantiene pipelines/ETL, ingesta, limpieza, orquestación, calidad de datos, y pone datos listos para consumo (analytics/ML). Optimiza almacenamiento, rendimiento y confiabilidad operativa.
- **Científico de Datos**: formula el problema analítico, explora datos, crea variables, entrena/evalúa modelos y comunica hallazgos. Se enfoca en inferencia/predicción y validación.
- **Arquitecto de Datos**: define la arquitectura global (capas, herramientas, estándares, gobierno, seguridad, modelado lógico/físico). Asegura que la plataforma sea escalable, mantenible y alineada al negocio.

3) ¿Cómo reduce dimensiones PCA?

PCA reduce dimensiones proyectando los datos a un nuevo sistema de ejes (componentes principales) que capturan la mayor variabilidad posible. En lugar de usar las variables originales, usa combinaciones de ellas que resumen la información, permitiendo:

- representar el dataset con menos dimensiones,
- disminuir redundancia/colinealidad,
- facilitar visualización (por ejemplo a 2D) y acelerar algunos procesos.

4) Diferencia entre importancia de variables y poder predictivo

- **Importancia de variables** suele referirse a una medida interna del modelo (por ejemplo, cuánto aporta una variable al reducir error o mejorar una partición). Depende del algoritmo y puede variar entre modelos.

- **Poder predictivo** mide qué tan bien una variable (o conjunto) ayuda a predecir el objetivo de forma verificable (por ejemplo, con métricas y validación). Una variable puede verse “importante” en un modelo específico, pero no necesariamente generaliza o mantiene desempeño fuera de muestra.