Explicació de les funcions de la simulació

Adrian Jiménez Franco

2025

Eines i entorn de simulació

La simulació s'ha desenvolupat íntegrament en **Python 3.11**, fent servir llibreries especialitzades per a l'anàlisi de xarxes de fluids (vegeu Taula 1). Les execucions es van dur a terme en un portàtil MSI Prestige 14 H B13UCX amb Windows 11 Pro, CPU Intel Core i7-13620H (10 nuclis/16 fils) i 16 GB de RAM, amb Python instal·lat a través de `conda`. Per garantir la reprodutibilitat, es va fixar la llavor aleatòria mitjançant `np.random.seed(42)` i es va capturar tota la sortida amb el mòdul `logging`.

Paquet	**Versió**	**Propòsit**
`pandapipes`	1.7.0	Modelatge hidràulic i tèrmic de xarxes de gas i líquids [**pandapipes**].
`networkx`	3.2	Anàlisi topològic (centralitat, camins mínims) [**networkx**].
`numpy`	1.26	Càlcul numèric vectoritzat i generació de nombres aleatoris.
`pandas`	2.2	Manipulació d'estructures tabulars (dataframes).
`scipy`	1.13	Funcions estadístiques (z–score, distribucions).
`tqdm`	4.66	Barra de progrés en bucles de simulació.
`logging`	stdlib	Traçabilitat i depuració dels processos.
`matplotlib` / `seaborn`	3.9 / 0.13	Visualització exploratòria i gràfics de validació.

Taula 1: Llibreries utilitzades en la simulació.

Configuració de la xarxa

Paràmetres estocàstics principals

La Taula 2 recull el diccionari `PARAMS`, utilitzat en tota la simulació.

Clau	Rang / Valor	Impacte
`prob_fuga`	0.8	Probabilitat que la simulació contingui fuites.
`num_fugas`	(1, 3)	Nombre de fuites si se'n genera almenys una.
`tasa_fuga`	(0.05, 0.10) kg/s	Cabal màssic afegit per cada fuita.
`consumo_normal`	$(1 \times 10^{-4}, 3 \times 10^{-4})$ kg/s	Demanda típica injectada en nodes de consum.
`ruido_presion`	$(1 \times 10^{-4}, 1 \times 10^{-3})$ bar	Desviació estàndard del soroll gaussià en sensors de pressió.
`prob_fallo_sensor`	0.02	Probabilitat que un sensor retorni la lectura sense soroll (simula fallada).
`num_consumos`	(4, 8)	Nombre de nodes amb demanda en cada simulació.
`prob_cierre_valvula`	0.03	Probabilitat de substituir un tram per una vàlvula tancada.

Taula 2: Paràmetres de control de la simulació.

Relació paràmetre-component

- **Variabilitat estructural**: `prob_cierre_valvula` altera la connectivitat; es comprova que el tancament no aïlli cap node (anàlisi de connexió amb `networkx`).
- **Comportament dinàmic**: `num_consumos` i `consumo_normal` defineixen la càrrega hidràulica base; les fuites es col·loquen en nodes de centralitat alta.
- **Realisme de sensor**: el parell `ruido_presion`–`prob_fallo_sensor` injecta soroll gaussià i fallades intermitents, mimant el rendiment irregular de sensors industrials.
- **Balanceig hidràulic**: `tasa_fuga` i `consumo_normal` apareixen explícitament a l'equació de continuïtat (**??**); l'algorisme de `pandapipes` recalcula pressions i cabals segons la fricció de Colebrook.

simulate_sensor_reading

La funció `simulate_sensor_reading` pretén reproduir de manera realista el comportament d'un sensor de pressió industrial, que no sempre proporciona valors perfectament nets i actualitzats. Per a cada instant de mesura, s'ha de decidir primer si el sensor pateix un "congelament" intermitent (amb probabilitat 2 %). En aquest cas, en lloc de retornar un valor `NaN` o un senyal d'error, la funció opta per enviar la mateixa lectura vàlida anterior, simulant així un error d'estació fixa: el sensor queda aturat i mostra la darrera mesura bona. Quan no es produeix aquest tipus de fallada, la funció genera un desviament aleatori segons una distribució normal centrada a zero, amb una desviació estàndard σ triada uniformement entre 0,0001 i 0,001 bar. D'aquesta manera, s'obté un soroll gaussià de rang variable, tal com entrenen gestors reals per marcar la precisió limitada dels sensors.

La funció `simulate_sensor_reading` permet incorporar dues fonts d'error molt habituals en dispositius IoT: el soroll inherent de la mesura i els talls ocasionals de senyal, fent que el conjunt de dades simulat reflecteixi millor el comportament dels sensors en condicions reals de camp.

create_network

La funció `create_network` construeix dinàmicament una xarxa de distribució de gas sintètica utilitzant `pandapipes` i `networkx`. En primer lloc, es crea una xarxa buida per a gas licuat (`fluid="lgas"`) i se li assigna un nom únic basat en l'identificador de simulació (`sim_id_for_name`). A continuació, es generen 38 nodes amb índexs consecutius de 100 a 137. Cadascun d'aquests "junctions" s'inicialitza amb una pressió de càlcul de 1.0 bar (`pn_bar=1.0`), una temperatura de referència (`tfluid_k=t_ext`, típicament 293.15 K) i una altitud nul·la. El node 100 s'utilitza com a font externa de pressió i temperatura fixes (`p_bar=p_ext` i `t_k=t_ext`), establint la condició de contorn per al càlcul hidràulic de la resta de la xarxa.

Acte seguit, es recorre la llista `pipes_data` (definida a la Taula 3), on cada fila conté el node d'origen, el node de destí, la longitud en metres i el diàmetre en polzades de la tuberia corresponent. Per a cada definició, es comprova que ambdós nodes existeixen; si no, es registra una advertència i s'omet la creació d'aquesta tuberia. En cas afirmatiu, es crida `pp.create_pipe_from_parameters` convertint la longitud a quilòmetres (`length_km = length/1000`) i el diàmetre a metres (`diameter_m = diameter*0.0254`). Cada canonada es subdivideix en 10 seccions (`sections=10`) per millorar la precisió numèrica dels càlculs, i es fixen 0.05 mm de rugositat (`k_mm=0.05`), valor típic de canonades d'acer comercials.

Després de crear totes les canonades, es construeix un objecte `networkx.Graph` anomenat `G_initial` que emmagatzema les connexions entre nodes a partir dels camps `from_junction` i `to_junction` de la Taula `net.pipe`. Aquest grafo és la base per identificar possibles tancaments de vàlvules sense trencar la connexió completa amb la font (node 100).

La següent fase consisteix a tancar aleatòriament algunes canonades per simular el tall de vàlvules. Per cada tuberia (iterant sobre `net.pipe.index`), es genera un nombre aleatori i, si és inferior a `prob_cierre_valvula` (3 %), s'intenta canviar aquesta canonada per una vàlvula tancada. Abans de substituir-la, es fa una còpia de `G_initial` i s'elimina l'arista corresponent a la tuberia en qüestió. Si el grafo resultant continua connectat i tots els nodes poden arribar de nou al node font (comprovat amb `nx.is_connected` i `nx.has_path`), la canonada es retira i es crea una vàlvula tancada entre els mateixos nodes (`opened=False`). Aquest procés garanteix que mai no quedin nodes aïllats, mantenint la viabilitat hidràulica de la xarxa. Finalment, la xarxa modificada es retorna com a objecte `net` de `pandapipes`, amb la topologia inicial més els talls de vàlvules aleatoris, llistos per a la següent fase de simulació.

add_consumo_y_fugas

La funció `add_consumo_y_fugas` s'encarrega de col·locar tant els consums normals com les fuites en la xarxa creada prèviament, tenint en compte criteris de connexió i de centralitat per a ubicacions realistes. En primer lloc, es construeix el conjunt `connected_nodes`,

que inclou tots aquells nodes que estan connectats a alguna canonada (encara que aquesta pugui tenir vàlvules tancades). D'aquesta forma s'eviten nodes aïllats: només considerem com a vàlids aquells junctios que apareixen en les columnes `from_junction` o `to_junction` de la Taula 3. A continuació, es filtra per eliminar el node font (l'ext_grid), de tal manera que el conjunt final `nodos_validos` només inclou nodes on realment pot haver flux de sortida, ja sigui per consum o per fuita.

Si no existeix cap node vàlid, es registra una advertència i la funció retorna dues llistes buides, ja que no té sentit continuar sense punts d'injecció. Un cop disposats els nodes vàlids, es determina aleatòriament el nombre de punts de consum, comprès entre 4 i 8 (o menys si hi ha pocs nodes disponibles). Es trien aquests nodes per a consum sense reemplaçaments (es fa un `np.random.choice` sense repetició). Per cada node seleccionat, es genera un consum $\dot{m}_{\text{consumo}}$ uniformement entre 0.0001 i 0.0003 kg/s i es crea un `sink` en `pandapipes` amb aquest valor. Al mateix temps, es guarda en la llista `consumos_info` un diccionari amb la informació del node, el tipus ("consumo") i la taxa assignada. Així obtenim un patró de demanda base semblant al d'una xarxa real, on un petit subconjunt de nodes repeters punts de descàrrega.

Després, per identificar els punts potencials de fuita, es construeix el grafo `G_temp` de connectivitat considerant totes les canonades actuals i també les vàlvules obertes. Això assegura que la topologia reflecteixi l'estat dinàmic del sistema: nodes que només estiguin connectats a través de vàlvules tancades no es comptabilitzen com a trajectes actius. Si aquest grafo resulta buit o no és connex, es registra un avís i la funció retorna només la informació de consum, sense afegir cap fuita. En un entorn desconectat, aplicar mesures de centralitat no seria significatiu, perquè no hi ha camins únics de flux des de la font cap a cada node.

Suposant que `G_temp` és connex, es calcula la centralitat d'intermediació (`betweenness_centrality`) de cada node. Aquesta mètrica assenyala quins junctios s'utilitzen més sovint en camins mínims, és a dir, quins són estructuralment crítics per al flux global. Es descarta qualsevol node ja usat per a consum i s'ordenen els candidats en funció del seu valor de centralitat, de més alt a més baix. A continuació, es pren un màxim de 15 nodes "més centrals" per formar `nodos_criticos_top`. Si, per algun motiu (excepció o xarxa desconnexa), no es pot calcular correctament la centralitat, es recórrer a un "fallback" que tria nodes vàlids aleatòriament fins a cobrir el màxim nombre de fuites permeses (fins a 3).

Una vegada definits `nodos_criticos_top`, es determina de manera estocàstica si la simulació generarà fuites, amb una probabilitat del 80% (`prob_fuga`). Si la prova aleatòria dóna positiu i hi ha nodes crítics disponibles, s'escullen entre 1 i 3 nodes (segons `num_fugas`) per simular fuites. Per a cada node seleccionat, es crea un segon `sink` amb una taxa $\dot{m}_{\text{fuga}}$ uniforme entre 0.05 i 0.10 kg/s, distincta i molt superior a la dels consumidors normals. A més, s'emmagatzema en `fugas_info` un registre amb el node, el tipus ("fuga") i la taxa corresponent. D'aquesta manera, les fuites sempre apareixen en nodes topològicament rellevants, reproduint el fet que, en una xarxa real, els punts de pressió més alts o més concorreguts solen ser els més vulnerables.

Finalment, la funció retorna dues llistes senzilles: `fugas_nodos`, amb els identificadors dels nodes on s'han afegit fuites, i `consumos_nodos`, amb aquells on s'han creat consums normals. Aquests resultats es fan servir posteriorment per etiquetar cada fila del *dataset* amb les variables d'interès: un node amb `fuga=1` assenyala presència d'una fuita real en

aquella simulació, mentre que `es_consumo_discreto=1` indica un node amb un consum fix regular.

simulate_network

La funció `simulate_network` s'encarrega d'executar el càlcul hidràulic de la xarxa i retornar, només en cas d'èxit, una instància de `pandapipes` amb els resultats disponibles. En primer lloc, es comprova que la xarxa tingui una *font externa* definida (`ext_grid`), ja que sense aquesta condició no es podria alimentar cap flux. Si aquest pas falla, es registra un missatge d'error i la funció acaba retornant `None`. Tot seguit, es verifica que hi hagi algun element de flux: o bé tuberies (`net.pipe`) o bé vàlvules obertes (`net.valve` amb `opened=True`). Si la xarxa no conté cap conducte per on pugui circular el gas, de nou es produeix un error i s'abandona la simulació.

Un cop confirmada la presència de flux, la funció realitza el primer pas de càlcul mitjançant el model de fricció de Nikuradse (`friction_model="nikuradse"`). Aquest mètode és més estable i convergeix amb menys iteracions, per això s'utilitzen toleràncies relativament relaxades ($\Delta P < 10^{-3}$ bar i $\Delta \dot{m} < 10^{-3}$ kg/s) i un límit de 200 iteracions. Si aquesta primera fase converteix correctament, es passa a un segon càlcul emprant el model de Colebrook (`friction_model="colebrook"`), amb toleràncies més estrictes (10^{-4} tant per a pressió com per a cabal) i un màxim de 300 iteracions, de les quals fins a 150 es poden dedicar a resoldre implícitament l'equació de Colebrook. D'aquesta manera, la solució aproximada obtinguda amb Nikuradse serveix de punt de partida per a un càlcul més precís que corregeix els desajustos de fricció en tubs reals.

Si, després de la segona fase, la xarxa encara no ha convergit, la funció llença una advertència indicant que la simulació no s'ha completat satisfactòriament i torna `None`. En canvi, si la xarxa convergeix, la funció retorna l'objecte `net` amb les taules `res_junction` i `res_pipe` plenes: aquestes contenen, respectivament, la pressió calculada a cada node i el cabal mésic en cada secció de tuberia. Aquest resultat és fonamental per a l'etapa següent, en què s'extreuen característiques per a l'aprenentatge automàtic, ja que només s'utilitzen dades físicament consistents. A més, qualsevol excepció inesperada durant el procés de `pipeflow` queda registrada com un missatge d'error amb informació bàsica, facilitant la depuració sense interrompre la generació en bloqueig de tot l'experiment.

calcular_caracteristicas

La funció `calcular_caracteristicas` pren com a input la xarxa ja calculada hidràulicament (l'objecte `net` de `pandapipes`), juntament amb les llistes de nodes on s'han assignat fuites i consumos. El seu objectiu és transformar els resultats bruts de pressió i cabal en un *DataFrame* estructurat, amb variables que capturen tant l'estat físic instantani de cada node com la seva posició topològica dins de la xarxa.

En primer lloc, s'assegura que `net` existeixi i hagi convergit; si la simulació no ha produit cap resultat, es retorna un `DataFrame` buit i es registra una advertència, evitant processar xarxes no vàlides. A continuació, la funció determina la temperatura de treball mitjana mirant si hi ha el camp `t_k` en `res_junction`. Si existeix, fa la mitjana de totes les temperatures de node; si no, recorre a la temperatura inicial de cada `junction`. A partir d'aquest valor, obté la densitat del gas mitjançant el mètode `net.fluid.get_density`,

amb un *fallback* de 0.8 kg/m^3 en cas d'error, que sol correspondre a un valor típic diferent.

Un cop calculada la densitat, el pas següent és determinar el cabal mésic a cada tuberia. Si `res_pipe` conté la velocitat mitjana `v_mean_m_per_s`, es recorre cada fila i es calcula l'àrea transversal $\pi(d/2)^2$ a partir del diàmetre que apareix en `net.pipe.diameter_m`. El producte de densitat, velocitat i àrea genera el cabal $\dot{m}$ en kg/s, que s'emmagatzema a la columna `mdot_kg_per_s`. D'aquesta manera, disposem a `res_pipe` tant de velocitats com de fluxos mésics reals, ingredients indispensables per estudiar l'equilibri de masses als nodes.

Per mesurar l'impacte de la topologia de la xarxa, es construeix un graf `G` de `networkx` que inclou totes les aristes corresponents a canonades existents i, si n'hi ha, a vàlvules obertes. Aquest graf permet, en la fase següent, calcular la centralitat d'intermediació (`betweenness_centrality`) i la distància de cada node fins a la font (el node de l'`ext_grid`). Si la gràfica no és completament connexa, s'identifica el component que conté la font i es calcula centralitat i distàncies només dins d'aquest subgraf, garantint que les mètriques sempre reflecteixin vies de flux vàlides.

Amb la densitat, el graf de connexions i els resultats de pressió disponibles, s'inicia el bucle que recorre cada node `j` de `net.res_junction`. Per a cada node, s'anota la pressió real `pr` i es genera una mesura sintètica `ps` cridant `simulate_sensor_reading(pr, time_step)`, que simula soroll i possibles fallades de sensor. Després, es sumen tots els cabals entrants i sortints per aquesta unió: per trobar el cabal entrant, es busquen les canonades que acaben al node j i es sumen els valors absolut de `mdot_kg_per_s`; de manera anàloga, s'acumulen els cabals sortints. A més, si el node té consums o fuites (definits en `net.sink`), es sumen aquests cabals. El *desbalance* de flux local es calcula com $|\text{inflow} - \text{outflow} - \text{cons}|$, i serveix com a indicador directe de pèrdua de massa (una fuga) quan es desvia del zero.

Per caracteritzar el perfil de pressió al voltant del node, es recopilen les pressions dels veïns (els nodes adjacents en el graf `G`). A partir d'aquest conjunt de valors, s'obté la diferència entre la pressió pròpia `pr` i la mitjana local, ΔP, així com la desviació estàndard dels veïns. A més, es calcula l'*Z-score* local, que és la distància de `pr` respecte a la distribució de pressions del veïnat; si no hi ha veïns o la variància és nul·la, aquest valor s'estableix a zero.

Finalment, per a cada node es créen un seguit de variables que conformen l'estructura de la Taula de sortida. Entre elles hi trobem el grau de connectivitat (nombre de veïns), la centralitat i la distància a la font obtingudes al graf, un indicador binari de si aquell node contenia una fuita (`fuga`=1 si `j` està en la llista `fugas`, o 0 en cas contrari), i un indicador similar per als nodes de consum normal. També s'inclou l'hora del dia (`time_step%24`), el logaritme de la pressió real (per reduir l'efecte de valors elevats), la desviació estàndard en la pressió veïnal i el `zscore_presion_local`.

La sortida de la funció és un `pandas.DataFrame` amb una fila per a cada node i un total de 17 columnes:

- `sim_id`,
- `nodo`,
- `presion_real`,
- `presion_sensor`,
- `flujo_entrada_tuberia`,
- `flujo_salida_tuberia`,
- `delta_flujo_nodo`,
- `delta_presion_media_vecinos`,

- `centralidad`,
- `distancia_fuente`,
- `grado_conectividad`,
- `fuga`,
- `es_consumo_discreto`,
- `hora_dia`,
- `log_presion_real`,
- `std_presion_vecinos`,
- `zscore_presion_local`.

En conjunt, aquestes variables encapsulen informació física (pressió, flux) i estructural (centralitat, distància), juntament amb factors de soroll i variabilitat temporal, oferint un conjunt de característiques completes per entrenar algorismes de detecció de fuites en xarxes de gas. Si la simulació no ha convergit o no hi ha resultats vàlids, la funció retorna un *DataFrame* buit i registra un avís en el `logging`, permetent continuar amb la generació massiva de dades sense interrompre tot el procés.

generar_dataset_realista

La funció `generar_dataset_realista` coordina tot el flux de treball necessari per crear un conjunt de dades sintètic de múltiples escenaris de xarxes de gas amb fuites. El seu objectiu és produir, en cada hora simulada, cent xarxes independents i repetir aquest procés durant vint-i-quatre hores per assolir un total de 2.400 simulacions. Cada simulació comprèn quatre passos fonamentals: construcció de la xarxa, assignació de consums i de possibles fuites, càlcul hidràulic de pressions i flujos, i extracció de característiques per a l'exportació final.

En primer lloc, demanem dos paràmetres claus: `num_muestras_por_hora`, que sol ser 100, i `tiempo_max_horas`, típicament 24. També indiquem el nom del fitxer de sortida, per exemple `"dataset_fugas_gas_simulacion.csv"`. A mesura que avança la simulació, es manté un recompte de les xarxes intentades (`simulaciones_intentadas`) i de les que han generat dades vàlides (`simulaciones_exitosas`). Els resultats parcials s'acumulen en una llista `datos_acumulados`.

El bucle extern recorre cadascuna de les vint-i-quatre hores. Dins de cada hora, un bucle intern repeteix cent vegades el procés de creació d'una nova xarxa: primer es crida `create_network` amb un identificador únic (convertit a cadena) que fixa el nom intern de la simulació. A continuació, s'invoca `add_consumo_y_fugas` per etiquetar nodes amb consums normals (quatre a vuit nodes triats aleatòriament) i, amb una probabilitat del 80%, generar d'una a tres fuites en nodes d'alta centralitat. Amb aquesta topologia de flux, incloent canonades, vàlvules tancades, consums i fuites, s'executa finalment `simulate_network`, que realitza el càlcul hidràulic amb Pandapipes. Si la xarxa convergeix, la funció retorna l'objecte `net` amb els resultats de pressió i velocitat. En cas contrari, es descarta aquella instància i es registra una advertència.

Quan `simulate_network` confirma la convergència, es passa a `calcular_característiques`, que extreu, node a node, un conjunt de 17 atributs: pressió real, pressió simulada amb soroll, cabals entrants i sortints, desbalance local de flux, diferències i desviacions de pressió respecte al veïnat, centralitat d'intermediació, distància al node font, grau de connexió, indicadors binaris de fuites i consumos, hora del dia, transformació logarítmica de la pressió real, dispersió local (desviació estàndard dels

veïns) i `zscore_presion_local`. Aquesta matriu de característiques es retorna com un `DataFrame` que s'afegeix a `datos_acumulados`. Cada instància completada s'incrementa en `simulaciones_exitosas`.

Per garantir robustesa, cada sis hores (és a dir, quan `t_hora + 1` és múltiple de sis), se salva en disc una versió parcial del *dataset*. Així, en cas d'interrupció o error crític, només es perdrien com a molt sis hores de processament. Si l'usuari prem Ctrl +C, es captura la interrupció amb `KeyboardInterrupt` i es procedeix a escriure el que s'hagi acumulat fins aquell moment. En cas que aparegui qualsevol altra excepció greu dins de la fase de generació, es registra la informació d'error —incloent traza— sense aturar completament el procés fins arribar al bloc `finally`.

En l'etapa final, si s'han recollit dades en `datos_acumulados`, es concatena tota la llista en un únic `DataFrame`. Abans de desar el resultat, es fa una neteja mínima: s'eliminen totes les files que tinguin `NaN` a les columnes essencials (`sim_id`, `nodo`, `presion_real`). Les taules de sensors amb valors de pressió faltants es poden donar per fallada simulada, però no volem perdre els registres sense un identificador o sense pressió real. Si després de treure aquestes files el *DataFrame* encara conté registres, es barregen les files de manera aleatòria (`sample(frac=1)`) per evitar qualsevol patró temporal o seqüencial que l'algorisme de ML pugui aprendre indegudament. Finalment, s'escriu el fitxer complet en la ruta indicada per `output_file`. Si el *DataFrame* ha quedat buit, es registra una advertència que indica que no hi ha dades guardades.

Quan es finalitza tot el procés, es mostren per pantalla algunes estadístiques resum: el nombre total de files generades, el nombre de simulacions úniques i la distribució percentual de la variable `fuga`, així com de la variable `es_consumo_discreto`. Aquestes mètriques confirmen el balanç entre casos "amb fuga" i "sense fuga", i entre nodes que actuen només com a punts de consum normal. El fitxer resultant contindrà, per a cada hora i per a cada node de cada simulació, totes les característiques necessàries per entrenar i avaluar models de detecció de fuites sota condicions reals de soroll, variabilitat topològica i dinàmica de consum.

Un cop completat el bucle de generació i acumulació de totes les simulacions (tal com mostra el diagrama de flux de la Figura **??**), es construeix una base de dades realista d'un circuit de gas. Aquest procés dona lloc a una base de dades sintètica completa i variada, amb més de 90.000 instàncies etiquetades segons la presència o absència de fuita, així com les variables generades per a la seva detecció. A la Taula 4 es presenta la descripció d'aquestes variables.

ID Branca	Node Inici	Node Fi	Longitud (m)	Diàmetre (polzades)
0	100	101	216.1	6
1	101	102	199.9	6
2	102	106	98.1	8
3	105	106	78.9	3
4	104	105	139.9	3
5	101	104	92.0	4
6	103	104	119.9	3
7	100	103	81.1	8
8	100	120	135.9	6
9	120	121	20.1	4
10	121	134	93.9	4
11	122	121	75.9	4
12	122	123	78.0	6
13	122	126	153.0	4
14	123	124	70.1	6
15	124	125	63.1	6
16	125	126	110.0	3
17	126	127	60.0	3
18	126	129	96.0	4
19	134	127	96.9	4
20	125	128	100.9	4
21	128	129	73.2	4
22	128	131	89.9	3
23	129	130	67.1	4
24	127	130	92.0	4
25	130	137	95.1	4
26	137	133	67.1	3
27	132	137	98.1	3
28	133	135	267.9	3
29	135	134	60.0	3

ID Branca	Node Inici	Node Fi	Longitud (m)	Diàmetre (polzades)
30	135	136	70.1	3
31	103	136	49.8	8
32	136	107	53.0	8
33	107	108	223.1	4
34	108	109	221.9	4
35	106	109	93.0	8
36	109	118	15.8	6
37	118	117	111.9	6
38	116	117	88.1	6
39	117	119	84.1	6
40	115	116	70.1	6
41	113	115	150.0	6
42	108	113	103.9	4
43	110	113	252.1	6
44	113	114	89.9	4
45	107	110	118.9	8
46	110	111	152.1	12
47	111	112	56.1	12
48	132	131	50.0	8
49	114	116	30.0	6
50	104	108	100.0	4

Taula 3: Taula de propietats de la xarxa de gas, incloent-hi les longituds de les canonades, els diàmetres i els nodes d'inici i fi (1 polzada equival a 2,54 cm).

Nom de la variable	**Descripció**
`sim_id`	Identificador únic de la simulació. Permet agrupar totes les mostres generades en una mateixa execució completa.
`nodo`	Codi o índex del node ("junction") dins de la xarxa de gas. Correspon a l'identificador que s'utilitza en la topologia (per exemple, 100–137).
`presion_real`	Pressió calculada al node per la simulació hidràulica, en règim estacionari (model neumàtic).
`presion_sensor`	És la `presion_real` alterada per soroll gaussià o, en ocasions, coincident amb el valor real si ha ocorregut un "congelament" (fallada del sensor).
`flujo_entrada_tuberia`	Suma dels cabals màssics (kg/s) que entren al node a través de totes les canonades connectades cap a aquest node. Indica el flux total entrant.
`flujo_salida_tuberia`	Suma dels cabals màssics (kg/s) que surten del node a través de totes les canonades que en surten. Indica el flux total sortint.
`delta_flujo_nodo`	Desequilibri local absolut del node (Unitat: kg/s): $\lvert \texttt{flujo\_entrada\_tuberia} - \texttt{flujo\_salida\_tuberia} - \texttt{consumo} \rvert$. Reflecteix pèrdues o discrepàncies en el balanç de massa.
`delta_presion_media_vecinos`	Diferència entre la pressió del node i la pressió mitjana dels seus veïns immediats: $\Delta P = P_{\text{nodo}} - \frac{1}{\deg(\text{nodo})} \sum_{\text{veí}} P_{\text{veí}}$. Indica quant difereix la pressió pròpia respecte de l'entorn local.
`centralidad`	Centralitat d'intermediació (betweenness) del node en el graf operatiu (canonades + vàlvules obertes). Mesura quants camins mínims passen per aquell node i, per tant, quina importància té en l'arquitectura de flux.
`distancia_fuente`	Nombre mínim de salts (arestes) que separen aquest node del node font (`ext_grid`). Si el node és la font, el valor és 0; si és veí directe, 1; etc.

Nom de la variable	**Descripció**
`grado_conectividad`	Grau del node en el graf operatiu: nombre total de connexions directes (canonades o vàlvules obertes) que té aquell node.
`fuga`	Indicador binari de fuga: 1 si en aquell node s'ha generat un escape de gas (sink de fuga), 0 en cas contrari. Serveix com a etiqueta d'"anomalia" per a l'entrenament de models.
`es_consumo_discreto`	Indicador binari de consum normal: 1 si en aquell node s'ha creat un "sink" de consum fix, 0 en cas contrari. Permet distingir nodes que actuen com a demanda regular del sistema.
`hora_dia`	Hora de la simulació en què s'ha pres la mostra: valors cíclics entre 0 i 23. Necessari per atendre possibles patrons estacionals en el comportament de la xarxa.
`log_presion_real`	Logaritme natural de la `presion_real` (s'hi afegeix un desplaçament de 10^{-6} per evitar log(0)). Útil per reduir l'asimetria de la distribució de pressions en l'entrenament.
`std_presion_vecinos`	Desviació estàndard de les pressions en els nodes veïns immediats del node considerat. Reflecteix la dispersió local de valors i ajuda a detectar situacions anòmales.
`zscore_presion_local`	Z-score de la pressió en el node respecte al conjunt format pel propi node i els veïns immediats: $$z = \frac{P_{\text{nodo}} - \mu_{\{\text{nodo}\cup\text{veïns}\}}}{\sigma_{\{\text{nodo}\cup\text{veïns}\}}},$$ on μ i σ són la mitjana i la desviació estàndard d'aquest conjunt.

Taula 4: Descripció de les variables de la base de dades de simulació.