Roger Salla Sendra

Disseny i implementació d'un sistema de taules mestres per a especificacions de vehicles

TREBALL DE FI DE GRAU

Dirigit per Dr. Sergio Gómez Jiménez

Grau en Enginyeria Matemàtica i Física

UNIVERSITAT ROVIRA i VIRGILI

Tarragona

2025

Agraïments

En primer lloc, voldria expressar el meu agraïment a tots els professors i professores que m'han acompanyat al llarg de la meva trajectòria acadèmica, des de l'escola fins a la universitat. Gràcies per haver contribuït al meu creixement personal i formatiu. En especial, al Sergio, per la seva dedicació i guia com a tutor d'aquest treball.

També vull agrair als meus companys d'equip a IDIADA, que m'han facilitat l'adaptació durant les pràctiques i m'han ofert un entorn de treball acollidor i estimulant. Menció especial a la Sara, l'Agnès i l'Alfonso, per l'acompanyament constant i el seu suport en tot moment.

A la meva família i als amics, gràcies per ser-hi sempre. Pel suport incondicional en els moments difícils i per celebrar cada victòria com si fos vostra.

I finalment, als amics de la universitat. Gràcies per aquests quatre anys compartint classes i tardes d'estudi, però també sopars i festes. Sense vosaltres, aquesta etapa no hauria estat el mateix.

Resum

Aquest projecte té com a objectiu el disseny i implementació d'un sistema automatitzat per la creació i manteniment d'unes taules mestres de marques i models de vehicles. La solució desenvolupada cobreix totes les fases del cicle de vida de la dada: des de la construcció d'un procés ETL completament automatitzat, desplegat a la plataforma AWS, fins a la seva visualització mitjançant eines de BI com Qlik.

El procés complet es divideix en tres eixos principals:

- Normalització de marques i models: Resol el repte de netejar i estandarditzar aquesta informació creant un mestre de marques i models de vehicle, tenint en compte l'alta variabilitat en la nomenclatura. Per abordar aquest problema, s'ha fet ús combinat d'algorismes semàntics i models de llenguatge de grans dimensions (LLM), amb l'objectiu de detectar coincidències, unificar registres i construir una jerarquia consistent entre models, submodels i subsubmodels.
- ETL i automatització: S'executen tres funcions Lambda en cadena que s'encarreguen de llegir, comparar i transformar les dades. Aquestes funcions s'activen cada quatre mesos mitjançant AWS Step Functions.
- Cas d'ús: Es neteja una base de dades corporativa real i es dissenya un *dashboard* amb Qlik per avaluar el rendiment i la qualitat del procés.

A més, ha permès l'aplicació de coneixements adquirits en el grau d'Enginyeria Matemàtica i Física, especialment pel que fa a l'anàlisi crítica, el tractament massiu de dades i la resolució sistemàtica de problemes complexos.

Paraules clau: taules mestres, automatització, ETL

Resumen

Este proyecto tiene como objetivo el diseño e implementación de un sistema automatizado para la creación y mantenimiento de unas tablas maestras de marcas y modelos de vehículos. La solución desarrollada cubre todas las fases del ciclo de vida del dato: desde la construcción de un proceso ETL completamente automatizado, desplegado en la plataforma AWS, hasta su visualización mediante herramientas de BI como Qlik.

El proceso completo se divide en tres ejes principales:

- Normalización de marcas y modelos: Resuelve el reto de limpiar y estandarizar esta información creando una tabla maestra de marcas y modelos de vehículos, teniendo en cuenta la alta variabilidad en la nomenclatura. Para abordar este problema, se ha hecho uso combinado de algoritmos semánticos y modelos de lenguaje de gran tamaño (LLM), con el objetivo de detectar coincidencias, unificar registros y construir una jerarquía coherente entre modelos, submodelos y sub-submodelos.
- ETL y automatización: Se ejecutan tres funciones Lambda en cadena que se encargan de leer, comparar y transformar los datos. Estas funciones se activan cada cuatro meses mediante AWS Step Functions.
- Caso de uso: Se limpia una base de datos corporativa real y se diseña un *dashboard* con Qlik para evaluar el rendimiento y la calidad del proceso.

Además, ha permitido la aplicación de conocimientos adquiridos en el grado de Ingeniería Matemática y Física, especialmente en lo referente al análisis crítico, el tratamiento masivo de datos y la resolución sistemática de problemas complejos.

Palabras clave: tablas maestras, automatización, ETL

Abstract

This project aims to design and implement an automated system for the creation and maintenance of master tables of vehicle brands and models. The developed solution covers all stages of the data lifecycle: from building a fully automated ETL process, deployed on the AWS platform, to its visualization using BI tools like Qlik.

The entire process is structured around three main axes:

- Brand and model normalization: It addresses the challenge of cleaning and standardizing this information by creating a master table of vehicle brands and models, considering the high variability in naming conventions. To tackle this, a combination of semantic algorithms and large language models (LLMs) is used to detect matches, unify records, and build a consistent hierarchy between models, submodels, and sub-submodels.
- ETL and automation: Three chained Lambda functions are executed to read, compare, and transform the data. These functions are triggered every four months using AWS Step Functions.
- Use case: A real corporate database is cleaned and a Qlik dashboard is designed to evaluate the performance and quality of the process.

Furthermore, it has enabled the application of knowledge acquired during the Mathematical and Physical Engineering bachelor's degree, particularly in critical analysis, large-scale data handling, and the systematic resolution of complex problems.

Keywords: master tables, automation, ETL

Índex

1 Introducció **1**
1.1 IDIADA . . . 1
1.2 Organització d'IDIADA . . . 1
1.3 Digital Business . . . 2
1.4 Smart Workflow . . . 3
1.5 Data Analytics . . . 3

2 Motivacions del projecte **4**
2.1 Context . . . 4
2.2 Diccionari mestre de marques i models de vehicles . . . 5
2.2.1 Objectius . . . 6
2.2.2 Requeriments . . . 6
2.2.3 Antecedents . . . 7

3 Eines usades **8**
3.1 Python . . . 8
3.2 SQL . . . 9
3.3 Distància de Jaro-Winkler . . . 9
3.4 Model LLM . . . 10
3.5 Amazon Web Services (AWS) . . . 11
3.6 DBeaver . . . 11
3.7 Qlik . . . 11

4 Desenvolupament del projecte **12**
4.1 Generació de la primera versió del mestre . . . 12
4.1.1 Selecció de la base de dades de referència . . . 12
4.1.2 Tria de l'algorisme semàntic . . . 15
4.1.3 Gestió de les marques . . . 15
4.1.4 Gestió dels models . . . 21
4.1.5 Diagrama de flux . . . 28
4.1.6 Revisió processament models i nou *pipeline* . . . 31
4.1.7 Emmagatzematge de les taules generades . . . 33
4.2 Automatització del procés de generació i actualització del mestre . . . 36
4.2.1 *State Machine* . . . 36
4.2.2 1a Lambda. Carregar les dades . . . 37
4.2.3 2a Lambda. Control de versions . . . 39
4.2.4 3a Lambda. Actualització dels mestres . . . 41

5 Validació dels resultats **44**
5.1 Cas d'ús: Neteja de la base de dades de Pampol Samples . . . 44
5.1.1 *Script* de neteja . . . 45
5.1.2 Anàlisi dels resultats . . . 46

6 Conclusions **64**

7 Consideracions ètiques i de responsabilitat social **66**
7.1 Igualtat 66
7.2 Medi ambient 66
7.3 Responsabilitat social 66
7.4 Ètica 67

Referències **68**

Índex de figures

1 Base de dades de la EEA. 'CO_2 *emissions from new passenger cars'*. . . 12
2 Diagrama de flux processament marques 29
3 Diagrama de flux processament models 30
4 Diagrama de flux processament models amb la nova *pipeline* 32
5 Visualització mestre marques a DBeaver 34
6 Visualització mestre models a DBeaver 34
7 Visualització mestre especificacions a DBeaver 35
8 *State machine* 36
9 Estructura *bucket* d'origen 37
10 Visualització *query* i taula obtinguda 45
11 Full del *dashboard* neteja marques. Full 1 47
12 1a full del *dashboard* neteja models. Full 2 48
13 2n full del *dashboard* neteja models. Full 3 49
14 Marques descartades 50
15 Marques més repetides 51
16 *Dashboard* filtrat per les marques amb més variabilitat. Full 1 52
17 *Dashboard* filtrat amb marca = 'MERCEDES-BENZ' i mètode = 'LLM'. Full 1 53
18 *Dashboard* filtrat amb mètode = 'LLM'. Full 2 54
19 Secció estudi procés LLM models. Full 3 55
20 ALFA ROMEO SUPER al mestre 55
21 ASTON MARTIN VANTAGE al mestre 56
22 Exemple neteja codis alfanumèrics 56
23 Neteja del HYUNDAI SANTA CRUZ 57
24 Secció estudi procés NOT CONSIDERED models. Full 3 57
25 Exemple models omplerts amb codis alfanumèrics 58
26 Exemple models sense sentit 59
27 Exemple d'entrades mal categoritzades 59
28 Exemple de models no registrats al mestre 60

Índex de taules

1 Taula d'especificacions i la seva gestió 4
2 Valors descriptius del vehicle de la base de dades de la EEA 14
3 Representació del *DataFrame* que correspondria al diccionari abans mostrat 22
4 Representació del *DataFrame* conservant la jerarquia del diccionari a dalt mostrat 27
5 Comparativa entre l'script original i el nou pipeline optimitzat per models 33
6 Taula de patrons trobats durant l'anàlisi de models 61

Índex de codis

1 *Query* de lectura 14
2 Filtratge de les dades 14
3 Preprocessament de marques 17
4 Mòdul context i objectiu general 18
5 Mòdul passos a seguir 19
6 Mòdul exemple 20
7 Mòdul format resposta 20
8 Comprovació existència marca 24
9 Comprovació existència exacte model 24
10 Jerarquització del model 25
11 Comprovació supermodel 26
12 Funció `flatten_dict` 28
13 Ús de *sets* per cercar registres afegits i comuns 40
14 Tractament de *id*'s comuns 41
15 Funció `dataframe_to_dict` 42
16 *Query* a la base de dades de Pampol Samples 44

1 Introducció

Aquest projecte de final de grau forma part de les pràctiques realitzades a l'empresa Applus+ IDIADA, concretament a l'equip de *Data Analytics* del departament de *Digital Business*. Aquest departament té com a funció principal el desenvolupament i la implementació de solucions digitals que donen suport als diversos departaments de l'organització, amb l'objectiu d'optimitzar processos i impulsar la transformació digital de l'empresa.

L'objectiu principal d'aquest treball és l'obtenció d'unes taules mestres de marques i models de vehicles, per tal de minimitzar la variabilitat i inconsistència en la forma en què un vehicle apareix descrit en les diferents bases de dades de l'empresa, ja que aquesta manca d'homogeneïtat dificulta, entre d'altres, l'anàlisi de dades i la integració entre sistemes.

Aquest projecte també ha tractat l'automatització de l'actualització dels mestres i un cas d'ús pràctic de neteja que ha permès avaluar l'eficàcia del mestre generat i demostrar el seu impacte positiu.

1.1 IDIADA

Applus IDIADA és una empresa global d'enginyeria especialitzada en el desenvolupament de vehicles, proves i homologació. Amb seu a Santa Oliva (Tarragona), ofereix serveis a la indústria de l'automòbil per millorar la seguretat, l'eficiència i el rendiment dels vehicles. L'empresa disposa d'una de les pistes de proves més avançades d'Europa i de laboratoris d'última generació per a assajos físics i virtuals.

L'empresa compta amb més de 3.400 professionals i una xarxa internacional de 61 filials i sucursals a 24 països, cosa que garanteix que els seus clients rebin solucions ràpides i personalitzades. [1]

Applus, que ja gestionava IDIADA des de 1999, ha estat adjudicatària del nou contracte per administrar el centre tecnològic de 351 hectàrees a Santa Oliva aquest mateix setembre de 2024. L'adjudicació s'ha fet per un import de 428 milions d'euros, gairebé el doble de la licitació inicial, i permetrà a Applus controlar el 80% de l'empresa durant 25 anys. La Generalitat de Catalunya mantindrà el 20% restant de l'accionariat. Aquesta operació suposarà uns ingressos totals de més de 800 milions d'euros per a la Generalitat, en la que es considera "la principal operació patrimonial de la història de la Generalitat".[2]

1.2 Organització d'IDIADA

L'organització d'IDIADA es divideix en diverses àrees especialitzades per cobrir totes les necessitats del desenvolupament i certificació de vehicles:

- **Vehicle Engineering** – Disseny, simulació i validació de vehicles i components estructurals.
- **Chassis & Body Development** – Desenvolupament de suspensions, direcció, frens i carrosseria.
- **Passive Safety** – Proves de xoc, sistemes de retenció i protecció dels ocupants.

- **Active Safety & ADAS** – Validació de sistemes d'assistència a la conducció i conducció autònoma.
- **Powertrain & Emissions** – Desenvolupament i certificació de motors de combustió, híbrids i elèctrics.
- **Electromobility & Battery Testing** – Assajos de bateries, motors elèctrics i sistemes d'alt voltatge.
- **Durability & Reliability** – Proves de resistència, fatiga i durabilitat en condicions extremes.
- **Homologation & Certification** – Certificació de vehicles segons normatives internacionals.
- **Aero & Thermal Management** – Anàlisi aerodinàmica i gestió tèrmica dels vehicles.
- **Connected & Automated Vehicles** – Recerca en comunicació vehicle-xarxa i conducció autònoma.
- **Digital Business** – Provisió d'un ecosistema d'eines i serveis modulars per millorar l'eficiència dels processos d'enginyeria i testing, permetent extreure coneixement de les dades.

1.3 Digital Business

El departament de *Digital Business*, també conegut com a *Digital Solutions*, d'Applus+ IDIADA es dedica a digitalitzar processos i integrar tecnologies per millorar l'eficiència en el disseny i validació de vehicles. Ofereix eines software i hardware per gestionar assaigs i recursos, així com serveis de dades i informació per suportar estudis de viabilitat i comparatives. El seu objectiu és acompanyar els clients en la transformació digital i en l'optimització de les operacions mitjançant la tecnologia.

Ofereixen solucions multiplataforma amb integració contínua i desenvolupament de serveis basats en el núvol. A més, gestionen i controlen les dades dins l'organització, garantint-ne l'accessibilitat, la seguretat i un ús eficient. Integren les dades amb els formats estàndard de la indústria i utilitzen *Machine Learning* (Aprenentatge Automàtic) i Intel·ligència Artificial per analitzar-les i extreure'n el màxim valor. [3]

Treballen en dues principals branques d'eines de software i hardware:

1. **Eines per a la gestió d'assajos i instal·lacions:** Eines de software basades en web que se centren en la gestió de mostres de vehicles, programació de tests, planificació de recursos, gestió d'estacions elèctriques i gestió de pistes de proves.
2. **Eines per a l'execució i l'anàlisi d'assajos:** Aplicacions de software dedicades a la comprovació i la validació de les prestacions del vehicle, la simulació i la definició d'objectius funcionals, incloent-hi informes preconfigurats d'alta qualitat i opcions de personalització.

Pel que fa als serveis de dades, ofereixen accés a continguts homogenis de dades al núvol, elaborats i analitzats per experts de l'empresa, donant suport així a estudis de viabilitat del mercat i d'avaluació comparativa.

1.4 Smart Workflow

La secció de Smart Workflow d'IDIADA es dedica a desenvolupar eines digitals que automatitzen processos i gestionen dades de manera eficient. Aquestes solucions estan dissenyades per facilitar la feina d'enginyers i tècnics, i això permet la integració amb processos existents mitjançant serveis API[1] i l'ús de formats de dades estandarditzats.

Entre les eines destacades es troba IRIS, una plataforma intel·ligent que proporciona coneixement tècnic regulador precís i estructurat per a la indústria de l'automoció. Aquesta plataforma ajuda les empreses a mantenir-se actualitzades sobre les normatives actuals i futures a escala mundial.

La secció de Smart Workflow es divideix en quatre equips. Tres d'ells es dediquen a la gestió de les aplicacions web que conformen la *pipeline* de treball, mentre que el quart equip ofereix suport en diferents projectes tant interns, per exemple creant *dashboards* de *monitoring* per les diferents aplicacions, com externs, com és el cas d'Euro NCAP o la DGT.

Les tres aplicacions són:

- **TELMA**, una eina destinada a la gestió dels assajos realitzats i recursos usats en aquests i també planificació de nous assajos (recursos materials i humans, mostres, pistes...).
- **DARWIN**, enfocada en la creació de catàlegs de mètriques mesurables i representacions gràfiques. Per exemple, permet definir el tipus de gràfica esperada (de barres, de línies, etc.) per a una relació velocitat vs. temps, establint prèviament com a mètriques la velocitat i el temps, juntament amb les seves unitats corresponents.
- **DNA**, una aplicació que facilita la creació de gràfiques a partir de la ingesta de dades extretes dels assajos, treballant amb fitxers en format JSON.

1.5 Data Analytics

L'equip de Data Analytics s'encarrega de donar suport a diferents clients, tant interns com externs. Entre d'altres, les seves funcions són:

- **Informes personalitzats**: Operacions PWT[2], dashboards externs
- **Dashboards de monitoratge**: TELMA, DARWIN, Desktop APPS, DNA
- **Servei de dades SPECS**: Gestió de dades d'especificacions de vehicles

[1] Una API, o interfície de programació d'aplicacions, és un conjunt de regles o protocols que permet a les aplicacions informàtiques comunicar-se entre si per intercanviar dades, característiques i funcionalitats [4].

[2] Powertrain, departament de l'empresa dedicat als productes i serveis per al desenvolupament de línies motrius de vehicles.

2 Motivacions del projecte

2.1 Context

L'estada de pràctiques s'ha dut a terme al departament de *Digital Business*, concretament a l'equip de *Data Analytics*, que recentment ha evolucionat des de la seva versió anterior, coneguda com a *BI Testing*. Aquest canvi s'ha impulsat amb l'objectiu d'ampliar el seu abast, passant de donar suport exclusivament a clients dedicats al testatge de mostres a oferir solucions més generalitzades i adaptades a un major rang de possibilitats.

Així i tot, les mostres continuen sent un dels principals eixos de treball de l'equip. Fins ara, les especificacions de les mostres de vehicles, pneumàtics, bateries, components, etc., que entren a IDIADA eren gestionades a Pampol Samples. Pampol Samples és l'aplicació mitjançant la qual es gestionen les entrades i sortides de mostres d'IDIADA (juntament amb les especificacions d'aquestes) i, fins ara, opera amb la seva pròpia base de dades.

En aquest context neix SPECS, un projecte que pretén convertir-se en el sistema mestre per a les especificacions. L'objectiu final és que l'aplicació de Pampol Samples pugui consumir directament les especificacions emmagatzemades en els mestres de SPECS.

Tanmateix, hi ha una excepció per a les especificacions marcades en vermell de la taula 1, que, per motius externs a aquest projecte, es continuaran gestionant directament a Pampol Samples. Tot i això, SPECS encara serà el sistema mestre, assegurant la coherència i centralització de la resta d'especificacions.

Especificació	Tipus	Gestionat per	Mestre
VIN	char	P. SAMPLES	SPECS
Maker	char	P. SAMPLES	SPECS
Model	char	P. SAMPLES	SPECS
Plate	char	P. SAMPLES	SPECS
Color	char	P. SAMPLES	SPECS
Engine application	list	SPECS	SPECS
TC or NA	list	SPECS	SPECS
Number of cylinders	list	SPECS	SPECS
Emission level	list	SPECS	SPECS
Insured value	numeric	SPECS	SPECS
Max Power [Kw]	numeric	SPECS	SPECS
Max Speed [rpm]	numeric	SPECS	SPECS
Max Torque [Nm]	numeric	SPECS	SPECS
Voltage Level	list	SPECS	SPECS
Integrated Inverter	char	SPECS	SPECS
Integrated Gearbox	char	SPECS	SPECS
Type	list	SPECS	SPECS

Taula 1: Taula d'especificacions i la seva gestió

Per les especificacions de mostra totalment gestionades per SPECS, hi ha disponible una API per tal que l'aplicació de Pampol Samples pugui consultar els valors per una mostra donada.

2.2 Diccionari mestre de marques i models de vehicles

A partir d'aquest moment quan es parli de mostres es parlarà de vehicles que entren a IDIADA sigui per ser testejats, homologats o provats en pista, ja que és el cas que interessa per aquest treball.

En aquest nou context, en què l'equip de Data Analytics comença a gestionar les especificacions de les mostres d'IDIADA, es planteja la importància de crear un mestre de marques i models de vehicles per tal de ser capaços de normalitzar aquestes dades.

L'empresa tracta amb una gran quantitat de mostres i, per tant, és necessari estandarditzar els noms dels vehicles amb els quals es treballa. Per posar en situació el lector: una marca i model es poden trobar escrites de forma molt diferent en diferents fonts i bases de dades. Això encara es veu més agreujat amb el fins ara sistema de gestió de mostres de IDIADA, on l'usuari omple els camps 'marca' i 'model' de forma totalment lliure, produint així una alta variabilitat en els noms introduïts i afegint-hi la possibilitat d'algun error tipogràfic (majúscules i minúscules, accents...).

Alguns exemples reals en marques són:

- **ASTON MARTIN:** ASTON MARTÍN, ASTON MATIN, AML
- **MERCEDES-BENZ:** MERCEDES, MB, MERCEDES_BENZ, MERVCEDES
- **BMW**: BMW I, Bayerische Motoren Werke, bmw, .BMW
- **AUDI**: AUDI SPORT, AUDI A4
- **LAMBORGHINI**: AUTOMOBILI LAMBORGHINI, LAMBORGINI

i en models:

- **SEAT LEON:** LEÓN, LEON 380, LEON DIESEL
- **NISSAN QASHQAI:** KASKAI, QASQAI, WASHQAI
- **AUDI Q3 BLACK EDITION:** Q3 BCK ED
- **BENTLEY BENTAYGA:** BENTAIGA, BENTYAGA

La forma més senzilla de solucionar això és tenir una taula mestra amb els noms que es faran servir a l'empresa. A tot vehicle que entri a l'empresa, sigui físicament o com a *input* en alguna base de dades se li assignarà un nom de la taula mestre.

2.2.1 Objectius

L'objectiu principal d'aquest projecte és establir una base de dades centralitzada i normalitzada per a les marques i models de vehicles gestionats a IDIADA. Els objectius específics són:

- **Creació d'un diccionari mestre**: Desenvolupar un repositori centralitzat de marques i models validats per a la seva consulta i ús en tots els sistemes d'IDIADA.
- **Normalització de les dades**: Estandarditzar els noms de marques i models de la base de dades actual de mostres per garantir coherència en totes les aplicacions i bases de dades.
- **Reducció d'errors tipogràfics**: Eliminar variacions no desitjades degudes a errors humans (majúscules, minúscules, accents, abreviatures).
- **Millora de la qualitat de les dades**: Assegurar que les dades siguin precises i consistents per evitar duplicitats i incoherències.
- **Facilitació de la interoperabilitat**: Facilitar el treball entre aplicacions de l'empresa com TELMA, DARWIN i DNA.
- **Automatització d'actualitzacions**: Establir protocols automàtics per afegir noves marques i models mantenint la coherència de les dades al llarg del temps.

2.2.2 Requeriments

Per assolir els objectius establerts, el sistema de taules mestres de marques i models ha de complir els següents requeriments:

- **Base de dades estructurada**: Creació d'una base de dades relacional que permeti l'emmagatzematge i la gestió eficient de marques i models.
- **Mecanismes de validació**: Implementació de lògiques per garantir la coherència i l'exactitud de les dades introduïdes.
- **Automatització d'actualitzacions**: Mecanisme automàtic per introduir noves marques i models de forma regular sense comprometre la integritat del sistema.
- ***Dashboard* d'administració**: *Dashboard* web per visualitzar la base de dades, revisió d'errors i control de la normalització.
- **Escalabilitat**: Capacitat del sistema per adaptar-se a un creixement en el volum de dades sense degradar-ne el rendiment.

Aquest conjunt de requeriments garantirà el correcte funcionament i la fiabilitat del sistema de taules mestres, facilitant la seva adopció en els processos d'IDIADA.

2.2.3 Antecedents

Malgrat l'aparent senzillesa de la idea, la seva implementació té un impacte significatiu en la gestió de dades dins d'IDIADA i, per tant, en el flux de treball. Per aquest motiu, durant els darrers anys s'han dut a terme diversos intents de crear aquests mestres i estandarditzar aquestes dades.

No obstant això, fins ara, aquestes iniciatives no han assolit l'efectivitat desitjada, generalment degut a limitacions tècniques i manca de temps per enfrontar-s'hi. Aquest projecte busca abordar aquests desafiaments i establir una solució definitiva i eficient.

Un primer intent pretenia crear aquest mestre tot fent *web-scraping*[3] de la pàgina d'Euro NCAP[4]. Aquesta idea va ser finalment descartada per un motiu: només hi apareixen els vehicles testats i, a part que no tots ho estan, en un testatge agrupen diferents vehicles amb les mateixes característiques.

Un intent diferent pretenia agafar bases de dades oficials públiques, concretament la de la NHTSA[5], la KBA[6] o la EEA[7], i fer el mestre agafant els únics (*uniques*) dels camps marca i model de cada una d'elles. Aquesta idea implícitament assumia que aquestes bases de dades mantenien un control de duplicats, no permetent que un model es pugui trobar escrit de dues formes diferents, cosa que, com es veurà durant desenvolupament d'aquest projecte, no és així, motiu que va fer fallit l'intent també.

Malgrat això, aquests intents van ser estudiats en l'inici d'aquest treball, per tal de buscar una base sòlida sobre la qual començar. L'avaluació d'aquestes opcions va permetre examinar-ne els avantatges i les limitacions, així com determinar fins a quin punt podien aportar valor al projecte. Aquest procés va ser clau per prendre decisions informades sobre l'enfocament més adequat.

[3] *Web-scraping* és una tècnica de programari o software informàtic per extreure informació dels llocs web.[5]

[4] L'*European New Car Assessment Program* és un programa europeu voluntari d'avaluació del rendiment de la seguretat dels cotxes. Publiquen els resultats a la seva web.

[5] *L'Administració Nacional de Seguretat del Trànsit a les Carreteres* (*National Highway Traffic Safety Administration*, NHTSA) és una agència del govern federal dels Estats Units, que forma part del Departament de Transport, centrada en les normes de seguretat dels automòbils.

[6] *L'Autoritat Federal de Transport per Carretera* (*Kraftfahrt-Bundesamt*, KBA) d'Alemanya és l'organisme encarregat de l'homologació i certificació CE de vehicles, components i accessoris

[7] *L'Agència Europea del Medi Ambient* (*European Environment Agency*, EEA) és una agència de la Unió Europea, la missió de la qual és la recollida, elaboració i difusió d'informació sobre la situació i l'evolució del medi ambient a escala europea.

3 Eines usades

3.1 Python

Python és un llenguatge de programació de propòsit general àmpliament utilitzat en anàlisi de dades, ciència de dades i desenvolupament de programari i la seva sintaxi permet als programadors expressar conceptes en menys línies de codi del que seria possible en llenguatges com C [6]. Una de les seves grans fortaleses és la seva sintaxi senzilla i la gran comunitat que ha generat múltiples llibreries útils.

En aquest projecte s'han utilitzat dues estructures de dades especialment rellevants:

- **Diccionaris**: són col·leccions no ordenades de parells clau-valor. Permeten un accés molt eficient a les dades a partir de claus úniques. Són útils per emmagatzemar informació estructurada, com ara paràmetres o mapes de valors.

  ```
  client = {
      "nom": "Anna",
      "edat": 34,
      "correu": "anna@example.com"
  }
  print(client["nom"])  # Mostra 'Anna'
  ```

  En aquest exemple, el diccionari `client` conté tres claus: `nom`, `edat` i `correu`, cadascuna associada a un valor.

  Els diccionaris també permeten operacions com l'actualització, la supressió de claus, iteracions, i comprovació de l'existència de claus, entre d'altres.

- **Sets**: són col·leccions desordenades d'elements únics. Serveixen per eliminar duplicats i efectuar operacions matemàtiques com unions, interseccions o diferències entre conjunts.

  ```
  clients_dia1 = {"Anna", "Marc", "Jordi"}
  clients_dia2 = {"Marc", "Laura", "Anna"}

  clients_recurrents = clients_dia1 & clients_dia2
  print(clients_recurrents)  # Mostra {'Marc', 'Anna'}
  ```

  En aquest exemple, s'utilitza l'operador `&` per obtenir els clients que han vingut dos dies seguits (intersecció). També es poden usar `|` per fer la unió i `-` per fer la diferència.

Les principals llibreries usades són:

- **Pandas** és una llibreria essencial per al treball amb dades tabulars. La seva estructura principal és el *DataFrame*, una taula bidimensional similar a un full de càlcul o una taula SQL. Els *DataFrames* permeten una manipulació eficient de les dades, com ara filtratge, agrupació, fusió, tractament de valors nuls, càlculs estadístics i molt més [7].
- **NumPy** és una llibreria per al càlcul numèric amb Python. La seva estructura fonamental és l'*array* n-dimensional, que permet operacions vectorials i matricials altament optimitzades. És la base de moltes altres llibreries d'anàlisi de dades i aprenentatge automàtic. També ofereix funcionalitats com àlgebra lineal, transformades de Fourier o generació de nombres aleatoris [8].
- **boto3** és el SDK[8] oficial d'Amazon Web Services (AWS) per a Python. Permet interactuar programàticament amb serveis d'AWS com S3, Lambda, DynamoDB, EC2, entre altres [9].

3.2 SQL

SQL (*Structured Query Language*) és un llenguatge estàndard de comunicació amb bases de dades relacionals. L'SQL es pot hostatjar (es pot utilitzar) dins d'altres llenguatges de programació. La principal característica d'aquest llenguatge és la seva simplicitat, ja que amb pocs coneixements es poden fer consultes sobre una base de dades [10].

3.3 Distància de Jaro-Winkler

La distància *Jaro-Winkler* és una mètrica de cadena que mesura la similitud entre dues seqüències. És una variant de la mètrica de distància de Jaro (1989, Matthew A. Jaro) proposada el 1990 per William E. Winkler [11].

L'algorisme de *Jaro-Winkler* utilitza una escala de prefix p que dona puntuacions més elevades a cadenes que coincideixen des de l'inici per una llargada predefinida del prefix ℓ. La puntuació resultant es troba normalitzada entre 0 i 1.

Val a dir que, malgrat que aquesta distància és referida molts cops com una distància mètrica, no es pot considerar una mètrica en el sentit matemàtic, ja que no obeeix la desigualtat triangular. Per entendre la distància de *Jaro-Winkler*, primer s'ha d'entendre la de *Jaro*. L'única diferència és que la primera afegeix puntuacions més elevades a les coincidències des de l'inici a la puntuació atorgada per la segona.

Donades dues cadenes s_1 i s_2, la similitud de *Jaro* sim_j és

$$sim_j = \begin{cases} 0 & \text{si } m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & \text{altrament} \end{cases}$$

[8]Un equip de desenvolupament de programari (*Software development kit*, SDK) és un conjunt d'eines de desenvolupament de programari que permet al programador crear aplicacions per a un sistema concret

On:

- $|s_i|$ és la longitud de la cadena s_i;
- m és el nombre de caràcters coincidents;
- t és el nombre de transposicions.

Dos caràcters de s_1 i s_2 es consideren coincidents només si són el mateix i no es troben més lluny que $\lfloor \frac{max(|s_1|,|s_2|)}{2} \rfloor - 1$ caràcters. El nombre de transposicions és el nombre de caràcters coincidents que no estan en l'ordre correcte dividit per dos.

Com s'ha comentat prèviament, la similitud de *Jaro-Winkler* utilitza una escala de prefix p que dona més pes a les cadenes que coincideixen des del principi. Donades dues cadenes s_1 i s_2, la similitud de *Jaro-Winkler* sim_w es calcula com:

$$sim_w = sim_j + \ell p(1 - sim_j)$$

On:

- sim_j és la similitud de *Jaro* entre s_1 i s_2;
- ℓ és la longitud del prefix comú al començament de les cadenes, amb un màxim de 4 caràcters;
- p és un factor d'ajust constant que determina quant s'incrementa la puntuació per tenir un prefix comú. Assegura no obtenir valors superiors a 1.

3.4 Model LLM

Els models de llenguatge de grans dimensions (*Large Language Models*, LLMs), com ChatGPT, són xarxes neuronals entrenades amb enormes volums de text i són capaços de comprendre, generar i transformar llenguatge natural i també poden aplicar conceptes tècnics.

Dins el departament de *Digital Business*, on s'estaven duent a terme les pràctiques, hi trobem l'equip de *Data Lab*, el qual està desenvolupant una IA, AIDA, amb un aprenentatge i finalitats dirigides a l'empresa. Llavors, se'ls va demanar fer ús del seu model en el nostre equip, al qual després d'una presentació de les motivacions i finalitats, van accedir.

També es va fer una sessió explicant com treballen aquests models en la base (quins recursos de tercers fan servir, etc.), i com fer-lo servir: definir paràmetres d'entrada, de sortida, tipus de resposta, etc, i, finalment, com crear un bon *prompt* [9]. S'hi ha d'indicar l'entrada, la tasca a realitzar, quins passos ha de seguir per completar la tasca i quin és el resultat esperat. També és recomanable afegir un exemple amb una entrada i quina seria la sortida esperada.

[9] Un prompt és una instrucció, pregunta, sol·licitud o text que es dona a un programa d'intel·ligència artificial perquè generi un resultat.

3.5 Amazon Web Services (AWS)

Amazon Web Services (AWS) és una plataforma de serveis al núvol que ofereix recursos informàtics, d'emmagatzematge i eines de processament distribuïdes. En aquest projecte, s'han utilitzat diversos serveis d'AWS per automatitzar processos, gestionar dades i executar funcions de backend de manera escalable i eficient [12].

A continuació, es descriuen els serveis emprats:

- **Lambda** és un servei de computació sense servidor que permet executar funcions en resposta a esdeveniments, sense haver de gestionar cap infraestructura. És ideal per a tasques curtes, puntuals i automatitzades.
  - S'ha utilitzat per processar fitxers de manera automàtica quan s'incorporaven nous objectes al *bucket* S3.
  - La funció es desplega fàcilment amb boto3 i es pot vincular a altres serveis com Step Functions o S3.
- **S3** és un sistema d'emmagatzematge d'objectes al núvol, altament disponible i escalable.
- **Step Functions** és un servei d'orquestració que permet coordinar diversos serveis d'AWS mitjançant fluxos de treball definits per estats. Ha estat essencial per definir fluxos seqüencials o paral·lels entre diferents lambdes.

La combinació d'aquests serveis ha permès implementar un sistema modular, automatitzat i fàcilment escalable. A més, gràcies a la integració amb la llibreria `boto3`, s'ha pogut controlar i desplegar programàticament tot l'entorn des de Python.

3.6 DBeaver

Aquest programari ofereix una interfície gràfica completa que facilita la interacció amb diverses bases de dades *SQL* i *NoSQL*, incloent-hi les de RDS [13].

- **Connexió directa** amb RDS mitjançant els paràmetres de connexió proporcionats per Amazon (*endpoint*, port, nom d'usuari i contrasenya).
- **Exploració visual** de l'estructura de la base de dades, cosa que permet navegar fàcilment per les taules, atributs i altres objectes de la base de dades.
- **Editor SQL**, el qual ha facilitat l'execució de consultes sobre les dades.
- **Visualització de dades** en format tabular, amb opcions de filtratge i ordenació.

3.7 Qlik

Qlik és una eina de *Business Intelligence* (BI) que permet la creació de visualitzacions interactives, *dashboards* i anàlisi de dades de manera intuïtiva. S'ha utilitzat per representar gràficament els resultats obtinguts. Qlik destaca per la seva capacitat d'exploració associativa i la seva ràpida resposta a consultes sobre grans volums de dades [14].

4 Desenvolupament del projecte

4.1 Generació de la primera versió del mestre

Tot i que els objectius del projecte estaven orientats a obtenir un resultat final amb un procés completament automatitzat, la primera fase del treball es va centrar a desenvolupar una primera versió del mestre, que actuaria com a punt de partida per a les iteracions posteriors. Aquesta versió inicial no només va servir per definir i validar els diferents enfocaments proposats, sinó que també va permetre identificar les principals dificultats i àrees de millora, essent clau per assegurar l'èxit de les fases següents del projecte.

4.1.1 Selecció de la base de dades de referència

El primer pas necessari era el d'escollir quina base de dades es faria servir com a base per la creació del mestre. Com s'ha mencionat en l'apartat d'*antecedents*, ja s'havien fet diversos intents per crear aquests mestres a partir d'algunes bases de dades, així que els avantatges i problemes que comportava l'ús de cadascuna d'elles eren fàcils d'estudiar. La decisió final va ser usar les de la EEA, disponibles a la seva web. Aquesta agència proporciona informació detallada sobre les emissions de diòxid de carboni (CO_2) dels vehicles nous matriculats a la Unió Europea, així com altres dades tècniques associades a cada model. La decisió d'utilitzar aquesta font com a base de dades principal es va fonamentar en diversos motius clau.

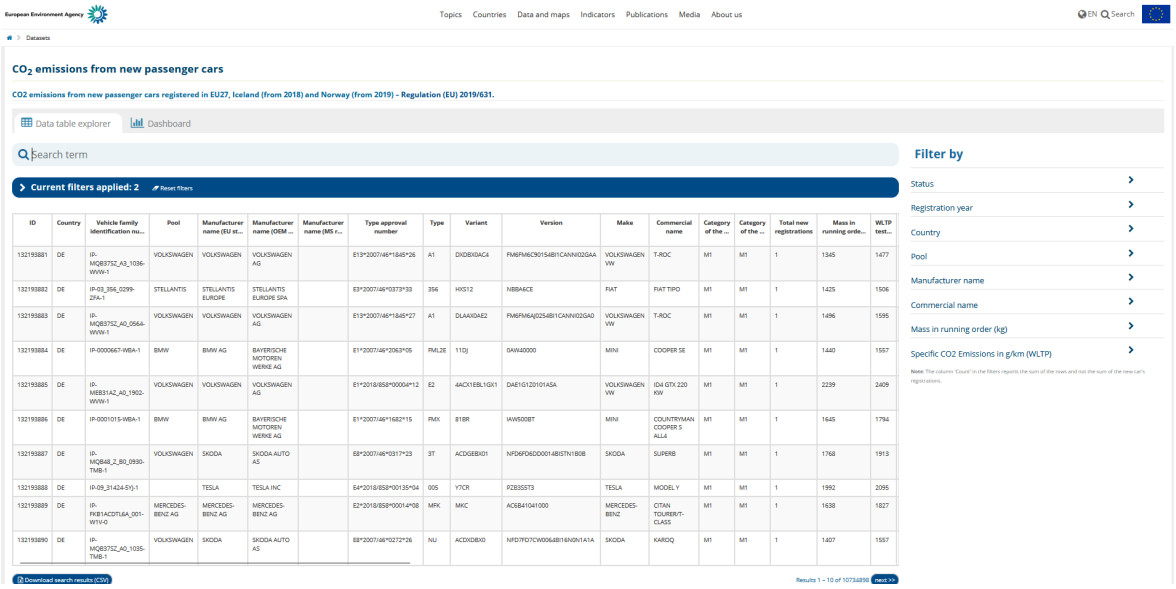


Figura 1: Base de dades de la EEA. 'CO_2 *emissions from new passenger cars*'.

En primer lloc, es tracta d'una font oficial i de gran fiabilitat, gestionada per un organisme europeu de referència en matèria de medi ambient i regulació. Aquesta credibilitat assegurava que les dades disponibles eren consistents, actualitzades i verificades, fet imprescindible per a garantir la qualitat del projecte. En segon lloc, la base de dades de la EEA conté una cobertura molt àmplia de marques i models de vehicles disponibles al mercat europeu, la qual cosa la feia especialment atractiva.

A més, un altre aspecte rellevant era que els noms de marques i models estaven en anglès, fet que facilitava la seva compatibilitat amb altres bases de dades internacionals. Per exemple, per la base de dades de la KBA (alemanya), es trobaven alguns noms escrits en alemany.

La base de dades de la NHTSA presentava menys varietat de marques i models, per la qual cosa també va ser descartada.

Un altre dels avantatges de la base de dades de la EEA era la seva fàcil extracció, permetent a la mateixa pàgina descarregar de forma gratuïta els fitxers CSV amb les dades corresponents a cada any.

Extracció de dades i lectura al script

El *dataset* conté dades corresponents als vehicles matriculats des de 2010 fins a 2023 a Europa, per la qual cosa hom pot fàcilment imaginar la quantitat de registres per any que hi ha. La pàgina permetia descarregar-se un fitxer CSV per cada any i estem parlant de què cadascun d'aquests fitxers tenia $\approx$ 11 milions de registres i una mida propera als 2.5GB.

Com que aquestes dades ja havien estat tractades amb anterioritat, aquestes s'havien penjat a una base de dades PostgreSQL[10] de l'empresa, cosa que facilitava la lectura d'aquestes, ja que només s'havia de fer una crida SQL, veure secció: 3.2, per recollir la informació.

Tot el procés es va dur a terme amb *python*, veure secció 3.1, perquè conté llibreries àmpliament optimitzades com *pandas*, veure secció 3.1, o *NumPy*, veure secció 3.1, que permeten tractar amb grans volums de dades còmodament.

Malgrat això, l'alt volum d'informació feia que la lectura d'aquestes dades per part del script fos lenta i, a causa d'això, es va optar per seguir una estratègia de *chunks* (particions). Cada crida SQL (*query*[11]) llegia una de les particions i s'anaven concatenant els resultats filtrats. Aquesta *query* es mostra al codi 1.

Només es llegeixen les columnes d'interès, aquelles que contenen informació respecte a marca i model del vehicle, però també les que contenen les especificacions. Amb això, s'aconsegueix optimitzar el temps de lectura. A la taula 2 s'hi poden trobar les correspondències de nom de la columna i de la informació que conté.

Com és de preveure, el fet que cada matriculació produeixi un registre té com a destí una base de dades repleta de duplicats, i per això el procés de filtratge se centrava a eliminar aquests. El codi que filtrava aquests duplicats era ben simple, mostrat al codi 2.

Mitjançant aquest procés de filtratge, es reduïa el nombre de registres, per cada any, dels aproximadament 11 milions inicials a prop de $200,000$. Aquesta reducció era un pas fonamental per garantir la viabilitat i l'eficiència de l'execució de la resta del codi.

[10] PostgreSQL és un potent sistema de bases de dades relacionals d'objectes de codi obert.
[11] Una *query* és una sol·licitud precisa per extreure dades d'una base de dades o sistema d'informació.

```
query = f"""
    SELECT
        da."ID",
        da."Tan",
        da."T",
        da."Va",
        da."Ve",
        da."Mk",
        da."Cn",
        da."Status"
    FROM "Datos_P_EEA_2020" da
    WHERE da."Status"= 'F'
    ORDER BY da."ID" ASC
    LIMIT {chunk_size}
    OFFSET {offset}
"""
```

Codi 1: *Query* de lectura

Nom columna	Valor que conté
ID	ID del registre
Tan	Contrasenya d'homologació del vehicle[1]
T	Tipus
Va	Variant
Ve	Versió
Mk	Marca
Cn	Model
Status	Estat de la dada (provisional o final)

[1] El format de la contrasenya d'homologació europea es divideix en quatre parts separades per asteriscs: eX*Directiva*XXXX*XX. Cada model de vehicle disposa d'una contrasenya d'homologació que defineix les seves característiques tècniques.

Taula 2: Valors descriptius del vehicle de la base de dades de la EEA

```
columns_to_consider = [col for col in df.columns if col != 'ID']
df.drop_duplicates(subset=columns_to_consider, keep='last', inplace=True)
```

Codi 2: Filtratge de les dades

4.1.2 Tria de l'algorisme semàntic

En aquesta fase preliminar, es va dur a terme una anàlisi comparativa de diversos algorismes de similitud textual amb l'objectiu de determinar quin seria el més adequat per a la tasca. L'objectiu era trobar una eina que permetés comparar noms de marques i models de vehicles.

Entre els algorismes analitzats hi havia *Levenshtein Distance* [15], també conegut com a *edit distance*, que calcula el nombre mínim d'operacions (insercions, eliminacions o substitucions) necessàries per convertir una cadena de text en una altra. Tot i la seva simplicitat i la seva aplicació àmpliament estesa en problemes de coincidència de text, es va descartar perquè no prioritzava les coincidències al principi de la cadena, un aspecte molt rellevant en el context del projecte.

Un altre algorisme considerat va ser el *Cosine Similarity*, aplicat sobre vectors *TF-IDF (Term Frequency – Inverse Document Frequency)* [16]. Aquesta tècnica, habitual en processament del llenguatge natural, representa cada cadena com un vector en un espai multidimensional i calcula l'angle entre ells per determinar-ne la similitud. Tot i que és molt potent en contextos amb textos llargs o quan es treballa amb grans volums de documents, es va descartar perquè no era òptim per a cadenes curtes com els noms de marques o models. A més, aquest enfocament requeria una fase de preprocessament considerable i no aportava un avantatge clar respecte d'algorismes més lleugers.

Finalment, la decisió va ser utilitzar l'algorisme *Jaro-Winkler*, ja que aquest afavoreix explícitament les coincidències a l'inici de les cadenes de text. A més, ofereix una bona sensibilitat davant petites diferències ortogràfiques. Aquest algorisme queda explicat a la secció 3.3. El fet que doni puntuacions més altes a les cadenes que coincidissin des del principi permet tractar amb més exactitud casos com BMW / BMW I o MERCEDES-BENZ / MERCEDES-BENZ AMG.

4.1.3 Gestió de les marques

La primera lògica plantejada per abordar el problema va consistir a utilitzar un mestre ja existent, el ja mencionat mestre fet a partir de *web-scraping* de la pàgina d'Euro NCAP, i ampliar-lo amb les dades noves de la EEA. L'estratègia se centrava primer a buscar coincidències semàntiques entre aquest mestre i les noves dades mitjançant un algorisme de comparació de cadenes. La idea era identificar coincidències amb els valors del mestre i, posteriorment, definir com gestionar marques i models nous que no hi figuraven.

Pel que fa a les marques, es va observar que aquesta aproximació era prou robusta sempre que el mestre fos prou extens, atesa la menor variabilitat en la forma d'escriure els noms comercials. La majoria de diferències eren errors ortogràfics, majúscules o minúscules o alguna diferència com per exemple el cas dels models elèctrics de BMW, que en alguns registres apareixien amb el nom de marca BMW I en lloc del nom correcte BMW.

Inicialment, totes les marques que no es trobaven al mestre s'afegien de forma automàtica, sense una comprovació prèvia, assumint que la EEA feia prèviament la comprovació de duplicats, no permetent que una marca es trobés escrita de formes diferents.

Ràpidament, però, això va comportar problemes, per aquesta mencionada duplicitat de marques equivalents amb noms lleugerament diferents, com ara LAMBORGHINI i AUTOMOBILI LAMBORGHINI o l'existència de dades on al camp de la marca del vehicle s'hi trobava el nom de l'empresa propietària o compradora del vehicle. Aquest tipus de discrepàncies van exposar les limitacions d'un enfocament purament basat en algorismes semàntics.

Això va fer sorgir la idea d'explorar l'ús de models de llenguatge de grans dimensions (LLM[12]) per tractar casos on l'algorisme semàntic no arribés. Així doncs, es va definir la idea de contrastar les marques que apareixen a la EEA amb les del mestre del Euro NCAP amb l'algorisme semàntic primer i, les que no fessin *match*, contrastar-les amb un model LLM que permetés trobar coincidències entre noms escrits molt diferents, però que fan referència a la mateixa marca.

A més, aquest model LLM hauria de ser de gran ajuda també a l'hora de tractar amb casos on la marca no figura al mestre i s'ha d'escollir si és una marca vàlida i, per tant, s'hauria d'incloure en el mestre final o no.

El percentatge de *matchs* entre el mestre del Euro NCAP i el de les dades de la EEA era d'aproximadament un 80%. Pel que fa al 20% restant, incloïa casos com LAMBORGHINI / AUTOMOBILI LAMBORGHINI que clarament no eren tractables amb un algorisme semàntic. S'hauria de fer una crida al model LLM. Evidentment, l'objectiu era minimitzar al màxim el nombre de crides al model, per tal de minimitzar costos. Per tant, es va optar per tractar d'ampliar el mestre de marques manualment el màxim possible, tot recopilant informació d'internet [18, 19]. Amb això, el percentatge de coincidències va augmentar fins al 95%. Fet que va reduir dràsticament la necessitat de recórrer al model LLM.

Aquest 5% de marques restants s'enviava al model LLM amb un triple objectiu: determinar si es tractava realment d'una marca existent amb el nom escrit de forma molt diferent, fet que feia que la similitud no arribés al *threshold* marcat per l'algorisme semàntic i, per tant, que hagués estat descartada, si es tractava d'una marca nova (no existent al mestre), o bé si eren simplement casos de dades brutes o inconsistents.

En el cas de detectar una marca nova, es demanava al model no només que la identifiqués correctament, sinó també que retornés una versió normalitzada del seu nom. Per exemple, suposant que LAMBORGHINI encara no hi fos al mestre, davant dels casos LAMBORGHINI i AUTOMOBILI DE LAMBORGHINI, el model retornava únicament LAMBORGHINI, unificant així les entrades i millorant la coherència del conjunt de dades.

Malgrat la seva sorprenent capacitat de categorització i comprensió, l'ús del model LLM també presentava una certa incertesa operativa: el fet que el model treballi amb probabilitats i no retorni exactament el mateix resultat en cada crida, generava una sensació de poca fiabilitat, especialment durant les primeres fases d'ús, on encara no s'havia treballat i refinat prou el *prompt*. Tot i això, els resultats obtinguts demostraren ser notablement satisfactoris.

[12] *Un model de llenguatge de grans dimensions* (*Large language model*, LLM) és un tipus de model d'aprenentatge automàtic dissenyat per a tasques de processament de llenguatge natural com ara la generació de llenguatge. Els LLM són models lingüístics amb molts paràmetres i s'entrenen amb un aprenentatge autosupervisat en una gran quantitat de text [17].

Script per al processament de les marques

Procés semàntic

L'*script* s'inicia amb la generació d'una llista de totes les marques distintes presents al *dataset* de la EEA. A continuació, es duu a terme una primera fase de neteja manual (preprocessament) sobre alguns casos típics i recurrents, amb l'objectiu de reduir la variabilitat i, conseqüentment, minimitzar el nombre de crides necessàries al LLM.

```
makers_list = [
str(x).strip().upper() for x in list(df['Mk'].unique()) if str(x).strip() !=
↪  ''
]

for maker in makers_list:
    if maker == 'Škoda' or maker == 'ŠKODA':
        maker = 'Skoda'
    elif maker == 'VW':
        maker = 'VOLKSWAGEN'
    elif maker == 'FORD - CNG-TECHNIK' or maker == 'FORD-CNG-TECHNIK' or
    ↪  maker == 'FORD-CNG TECHNIK':
        maker = 'FORD'
    elif maker == 'DS AUTOMOBILES':
        maker = 'DS'
    elif maker == 'MITSUBISHI MOTORS CORPORATION' or maker == 'Mitsubishi
    ↪  Motors (Thailand) C' or maker == 'MITSUBISHI MOTORS THAILAND Ltd.':
        maker = 'MITSUBISHI'
    elif maker == 'HYUNDAI,GENESIS':
        maker = 'GENESIS'
    elif maker == 'MERCEDES AMG' or maker == 'MERCEDES-AMG':
        maker = 'MERCEDES-BENZ'
    elif maker == 'EGO':
        maker = 'E.GO'
    elif maker == 'MC LAREN':
        maker = 'MCLAREN'
    elif maker == 'AUTOMOBILI LAMBORGHINI':
        maker = 'LAMBORGHINI'
```

Codi 3: Preprocessament de marques

Tot seguit, per cada marca de la llista, es busca la millor coincidència amb algun dels noms del mestre amb l'ajut de l'algorisme semàntic. Per les marques, a partir d'una anàlisi exploracional, es va establir un llindar de similitud (*threshold*) fixat en 0.9. Valors més alts eren massa restrictius mentre que baixar-lo produïa acabar amb noms mal relacionats. Les marques que no troben cap coincidència amb una puntuació superior a aquesta, se'ls hi assigna una cadena buida ".

Tot seguit, es crea un *DataFrame* que fa de diccionari: una de les columnes conté la llista 'bruta' de noms i l'altre conté la llista 'neta'. En el següent pas, es processa cada fila del *dataset* original, tot corregint el nom en cas d'haver trobat una coincidència. Pels casos als quals se'ls hi havia assignat *"*, afegirem el nom 'brut' a una llista (`makers_not_merged`) per tal que siguin processats més tard pel model LLM. En una altra llista (`info_makers`), també guardarem la resta d'informació de la fila, per tal de tornar a relacionar-la més endavant.

Procés LLM

En aquest punt, es fa la crida al model LLM. Com a `input` se li envia una llista de noms 'bruts' i l'`output` esperat és una llista amb els noms 'nets'. És important fer referència a alguns aspectes claus del *prompt*, ja que l'estructuració d'aquest és un factor clau en la recerca d'un bon `output`.

Durant tot aquest treball, totes les crides al model LLM tindran una estructura dividida en diversos mòduls lògics, cadascun amb una funció concreta. L'estructura es repetirà i el que canviarà serà el contingut. Per evitar repeticions, l'estructura s'explicarà només un cop i els següents només es comentaran els canvis que es considerin imprescindibles.

Context i objectiu general

Introdueix el propòsit global: classificar noms de marques brutes comparant-les amb una llista mestra de marques correctes. També es defineixen clarament els dos inputs: `DIRTY_BRAND_LIST` i `MASTER_BRAND_LIST`.

```
'''
You are tasked with classifying a list of 'dirty' car brand names with their
corresponding master model names or, if you are sure there is no coincidence
with any brand of the master list, you can add it as a new brand name.
If that happens, you will have to add this brand to the master just in case
there comes the same brand you have just added differently written.
You will be provided with two input variables:

DIRTY_BRAND_LIST: This is a list containing 'dirty' car brand names.

MASTER_BRAND_LIST: This is a list of clean car brand names.
'''
```

Codi 4: Mòdul context i objectiu general

L'`input` com es veu és una llista amb els noms 'bruts' (aquells que mitjançant l'algorisme semàntic no ha aconseguit fer *match*) i una llista amb el mestre.

Llista de passos a seguir

En aquest bloc, s'explica mitjançant una seqüència de passos com ha de completar la tasca assignada.

```
'''
Follow these steps to complete the task:

1. Process the DIRTY_BRAND_LIST by iterating through each value.
2. Compare the dirty brand name with each clean model name in the
MASTER_BRAND_LIST.
3. Use string matching techniques such as lowercase comparison, removing
special characters, and checking for substrings to find the best match.
4. If a match is found, assign the clean brand name to that value.
5. If no match is found, use your knowledge to determine if there is a
coincidence but written so differently that with string matching techniques
it is not considered a match, or if it is a new brand name not specified
in the MASTER_BRAND_LIST.
6. If it is a match, assign the clean brand name to that value.
7. If it is a new brand name, add it to the master in order to be able to
use it in future cases and assign this name.
8. If you are not sure about if it is a new brand or it is just something
bad written with no sense, assign an empty string ('').
9. Create a Python array where each element corresponds to the selected
clean brand name (or '') for the dirty brand name in the same position
of the DIRTY_BRAND_LIST.
10. For each value, you must be 90 percent sure about your answer.
If you are not, answer '' as that value's answer.
'''
```

Codi 5: Mòdul passos a seguir

Se li demana iterar sobre la llista de noms 'bruts', tot buscant coincidències en el mestre. Si no troba cap *match*, ha de valorar si es tracta d'una marca nova o simplement d'una errata. Per acabar, se li demana que torni una llista amb els noms 'nets', ja sigui perquè ha trobat coincidència amb alguna de les marques del mestre o perquè l'ha considerat com una nova. Si el nom 'brut' es descarta, en aquella posició a la llista de noms nets s'hi trobarà una cadena buida ".

Exemple d'entrada i sortida

Aquest bloc final il·lustra clarament com hauria de funcionar tot l'anterior, deixant clar el format d'entrada i el de sortida. També demostra com es tracta un cas nou (KOENIGSEGG) i com es descarten entrades errònies (NHDFGHFG → ").

```
'''
Here's an example of what the input and output might look like:

if the input was:
DIRTY_BRAND_LIST = ["lamborghini", "Automobili Lamborghini",
"Automobile Lamborghini", "NHDFGHFG", "Koenigsegg", "Ford Thailand LTC.",
"Mercedes AMG", "ford"]
MASTER_BRAND_LIST = ["LAMBORGHINI", "FIAT", "FERRARI", "SKODA",
"MERCEDES-BENZ", "FORD", "DACIA", "JAGUAR", "JEEP", "HONDA",
"MERCEDES-BENZ"]

the output would be:
["LAMBORGHINI", "LAMBORGHINI", "LAMBORGHINI", "", "KOENIGSEGG", "FORD",
"MERCEDES-BENZ", "FORD"]
and saved in memory that "KOENIGSEGG" is also a brand name not specified in
master_brand_list in order to use it as a clean model in future cases.
'''
```

Codi 6: Mòdul exemple

Instrucció de format de resposta

S'estableix un format explícit per a la resposta.

```
'''
Provide your answer as a Python array inside <answer> tags. Each element in
the array should be a string representing the clean brand name (or "")
corresponding to the dirty brand name in the same position of the
DIRTY_BRAND_LIST.
'''
```

Codi 7: Mòdul format resposta

De nou, com en el cas del procés semàntic, es crea un *DataFrame* que fa de diccionari entre noms 'bruts' i 'nets'. En últim lloc, s'ajunta de nou el nom 'net' amb la resta d'informació que prèviament s'havia guardat a la llista `info_makers`.

En aquest punt, ja s'ha generat el primer mestre de marques, tot havent recollit manualment des d'internet algunes i havent-ne afegit de noves. El procés també ha permès netejar els noms de les marques que apareixen al *dataset* de la EEA, la qual cosa serà de gran utilitat per la següent part on es crearà el mestre de models.

4.1.4 Gestió dels models

En aquest cas, la tasca resultava significativament més complexa, per dues raons principals: (1) la gran varietat de maneres diferents d'escriure un mateix model, i (2) el fet que dins del camp `model` s'hi podien trobar no només el nom del model en si, sinó també el del submodel i fins i tot d'un possible subsubmodel. Un exemple paradigmàtic d'aquesta situació és el cas del *Citroen C4 Cactus Aircross*. En aquest exemple, dins la base de dades de la EEA, al camp `Commercial Name (Cn)` apareix la cadena de text: `C4 Cactus Aircross`.

Tot i això, és fàcil entendre que el model principal és `C4`, el submodel és `Cactus`, i el subsubmodel és `Aircross`. Aquesta estructura jeràrquica era molt rellevant, ja que permetia no només estandarditzar el nom del model principal, sinó també extreure'n informació més precisa i detallada. Per tant, l'objectiu no era només netejar les cadenes, sinó també dividir-les en components.

Com en el cas de les marques, totes les cadenes eren convertides a majúscules per tal d'evitar que diferències en capitalització provoquessin falsos negatius en les comparacions. Ara bé, degut a l'altíssima variabilitat dels noms dels models, no era viable una estratègia basada només en similitud semàntica o coincidència directa de caràcters. A més l'extracció d'un mestre a partir de fonts ja existents tal com es va fer per les marques era molt més complicada, degut a l'alt volum d'informació que s'hauria de recollir, la variabilitat de nomenclatura entre les fonts...

Per tant, el procés semàntic va desaparèixer per aquesta part. El mestre es generaria des de 0, a partir de les dades de la EEA, tot normalitzant-les i estructurant-les jeràrquicament tal com prèviament ja s'ha explicat. Per gestionar aquesta complexitat, es va optar per una estratègia basada en l'ús d'un objecte molt versàtil en *python*: els diccionaris.

Un *diccionari* a *python* és una estructura de dades que permet associar una clau (*key*) a un valor (*value*). Veure secció 3.1.

Un aspecte especialment interessant de l'objecte és que el valor d'una clau pot ser al seu torn un altre diccionari. Aquesta propietat permet establir una estructura jeràrquica que encaixa de manera natural amb la forma en què els noms dels models, submodels i subsubmodels apareixen en les dades.

Considerant aquesta possibilitat, es va decidir implementar una estructura de dades recursiva on la jerarquia fos representada de manera explícita. Quelcom semblant al següent:

```
diccionari = {
    "CITROEN": {"C4": {"CACTUS": {"AIRCROSS", "SPORT"}}},
    "LAND ROVER": {"RANGE ROVER"},
    "AUDI": {"R8": {"S LINE"}, "A4": {"LIMOUSINE", "AVANT"}}
}
```

L'objectiu final era construir una estructura similar a l'anterior i utilitzar-la per tal de *xocar* les dades brutes amb aquesta base jeràrquica per detectar coincidències, estandarditzar noms i descompondre correctament les cadenes de text en els seus components.

Per construir aquesta estructura, es va desenvolupar una funció que, per cada valor del camp `model` que es llegia de la base de dades, actualitzava el diccionari, afegint nous valors quan calgués. En aquesta primera passada, però, no s'aplicava cap mena de lògica de similitud, per la qual cosa, casos com (C4 CACTUS SPORT/C4 CACTUS SPT) o (AUDI BCK ED/AUDI BLACK EDITION) eren considerats cadenes completament diferents. Aquesta estratègia garantia que es capturava tota la variabilitat real present en el *dataset*, que posteriorment seria tractada.

La normalització es va dur a terme, de nou, amb el LLM. Es va configurar un agent diferent, el qual rebia com a `input` el diccionari generat prèviament, i procedia a analitzar-lo per capes, comparant si dos valors podien correspondre al mateix concepte malgrat estar escrits de manera diferent, és a dir, el que passava en els casos descrits a dalt.

Amb els resultats retornats per aquest agent, es fusionaven les branques del diccionari que el model havia identificat com a equivalents. Aquest procés també contemplava la possibilitat de duplicitats dins de les mateixes branques i s'encarregava de gestionar-los adequadament.

Com a resultat, s'obtenia un diccionari refinat i estructurat jeràrquicament amb tots els models, submodels i subsubmodels estandarditzats. En paral·lel, també es mantenia una llista amb els noms de models que s'anaven llegint de la base de dades original. Un cop obtingut el diccionari definitiu, es procedia a contrastar cada element d'aquesta llista amb l'estructura jeràrquica, amb l'objectiu de retornar una versió estandarditzada del nom complet.

El primer pas d'aquest procés fou l'estandardització dels noms. Per tal de facilitar aquesta tasca, es convertí el diccionari en un *DataFrame*. En aquest cas, cada fila representava un model diferent extret del diccionari. Això permetria una consulta ràpida i eficient per identificar qualsevol estructura modelística en les dades.

Per exemple, en el cas del diccionari mostrat abans:

BRAND	***MODEL***
CITROEN	C4
CITROEN	C4 CACTUS
CITROEN	C4 CACTUS AIRCROSS
CITROEN	C4 CACTUS SPORT
LAND ROVER	RANGE ROVER
AUDI	R8
AUDI	R8 S LINE
AUDI	A4
AUDI	A4 LIMOUSINE
AUDI	A4 AVANT

Taula 3: Representació del *DataFrame* que correspondria al diccionari abans mostrat

En aquest punt ja es generava el mestre de models, el qual contenia la marca i el model (degudament separat en model, submodel i subsubmodel). Tanmateix, com a extra vam guardar també informació de totes les especificacions d'un mateix nom comercial: contrasenya d'homologació, tipus, variant i versió.

Per a fer-ho, una funció processava cada una de les files del *DataFrame* original que contenia aquesta informació. Com en el cas de les marques, amb un algorisme semàntic es buscava la millor coincidència en el mestre que s'acabava de crear. El *threshold* per aquest cas era de 0.9. Fent-ho, aproximadament un 80% dels noms eren correctament recollits. Pel 20% restant, es va configurar un últim agent LLM que rebia els noms no classificats, la llista de models de referència i se li demanava classificar-los.

Finalment, només quedava partir els noms dels models en model, submodel i subsubmodels. Es va escriure una nova funció que rebia la llista de noms i el diccionari de referència anteriorment creat i partia els noms. Val a dir que com més gran sigui la base de dades entrant més bona serà la qualitat del resultat i més informació contindrà. Per això, aquest procés descrit es repetirà amb els CSV corresponents amb tots els anys existents en la base de dades.

Script per al processament dels models

Aquest *script* té com a inici el *DataFrame* obtingut després del processament de les marques. Per cada una de les files, es fa un preprocessament del nom del model. Primer de tot, elimina el nom de la marca si també apareix al camp del model. Després per certes marques li fa un tractament especial:

- BMW: Extreu la sèrie del model: (320d → 3 SERIES 320D)
- MERCEDES-BENZ: Extreu la classe del model (C 200 → C-CLASS 200)
- SEAT: Comprova que no es tracti d'un CUPRA (prèviament s'ha observat que en alguns casos apareix el nom de la marca SEAT i al model hi posa, per exemple, CUPRA LEON). Si es tracta d'un CUPRA, canvia el nom de la marca i l'elimina del model.
- FIAT: Comprova que no es tracti d'un ABARTH (prèviament s'ha observat que en alguns casos apareix el nom de la marca FIAT i al model hi posa, per exemple, ABARTH 500). Si es tracta d'un ABARTH, canvia el nom de la marca i l'elimina del model.

Tot seguit, afegeix a la llista `ar` tota la informació *raw* (sense modificar) i actualitza el diccionari `dictionary_models` amb la funció `refresh_dictionary_model_exact`. Aquesta funció s'encarrega de mantenir l'estructura jeràrquica entre model, submodel i subsubmodel.

Aquesta funció és un pilar fonamental d'aquest treball. Està dividida en diverses parts que cobreixen diferents casos i possibilitats a mesura que es processa un nou valor.

1. **Creació de marca si no existeix:** La primera comprovació que realitza és si la marca (`maker`) ja es troba present al diccionari. En cas que no sigui així, s'inicialitza una nova entrada per aquesta marca amb el model proporcionat com a primera branca. Revisar codi 8.

```
if maker not in diccionario:
    diccionario[maker] = {model: {}}
    return diccionario
```

Codi 8: Comprovació existència marca

2. **Verificació d'existència exacta del model:** Si la marca ja existeix, la funció comprova si el model ja està registrat com a clau principal. En cas afirmatiu, no es fa cap canvi i es retorna el diccionari inalterat. Revisar codi 9.

```
if model in list(diccionario[maker].keys()):
    return diccionario
```

Codi 9: Comprovació existència exacte model

3. **Comprovació si el model ha de ser submodel o subsubmodel:** Quan el model no coincideix exactament amb cap existent, la funció analitza si pot ser considerat com un submodel d'algun model ja registrat. Això es fa comprovant si el model actual comença amb el nom d'algun model existent. Si es troba una coincidència parcial, es defineix la part restant com a submodel, i es continua el procés a un nivell inferior per comprovar si encara pot ser subdividit en un subsubmodel. Si les diferents capes existeixen, es retorna el diccionari sense modificacions. Si alguna capa no existeix, es crea en el lloc corresponent. Revisar codi 10.

4. **Creació d'un model com a supermodel:** En el cas que no es trobi cap coincidència com a model, submodel o subsubmodel, es fa una darrera comprovació: si algun model ja existent podria ser considerat un submodel del nou model. Això s'identifica observant si el nom d'un model existent comença pel nou nom que es vol afegir. Si aquest és el cas, s'efectua una reestructuració del diccionari: el model original passa a ser un submodel del nou nom, que ara actua com a model principal (supermodel). Si cap de les comprovacions anteriors ha trobat una ubicació adequada per al nou model, simplement s'afegeix com una nova entrada sota la marca corresponent. Revisar codi 11.

```
modelfound = False
submodelfound = False
subsubmodelfound = False
for existing_model in list(diccionario[maker].keys()):
    if model == existing_model or model.startswith(existing_model + ' '):
        modelfound = True
        submodel = model[len(existing_model):].strip()
        model = existing_model
        if submodel:
            for submodelref in diccionario[maker][existing_model]:
                if submodel == submodelref or
                ↪  submodel.startswith(submodelref + ' '):
                    submodelfound = True
                    subsubmodel = submodel[len(submodelref):].strip()
                    if subsubmodel:
                        for subsubmodelref in
                        ↪  diccionario[maker][existing_model][submodelref]:
                            if subsubmodel == subsubmodelref:
                                subsubmodelfound = True
                                return diccionario
                        if not subsubmodelfound:
                            diccionario[maker][existing_model]
                                        [submodelref][subsubmodel] = {}
                            return diccionario
                    else:
                        return diccionario
            if not submodelfound:
                diccionario[maker][existing_model][submodel] = {}
                return diccionario
        else:
            return diccionario
if not modelfound:
    diccionario[maker][model] = {}
```

Codi 10: Jerarquització del model

```
models_to_move = []
for existing_model in list(diccionario[maker].keys()):
    if existing_model.startswith(model):
        submodel = existing_model[len(model):].strip()
        if submodel:
            models_to_move.append((existing_model, submodel))

if models_to_move:
    diccionario[maker][model] = {}
    for old_model, submodel in models_to_move:
        diccionario[maker][model][submodel] =
        ↪ diccionario[maker].pop(old_model)
else:
    diccionario[maker][model] = {}

return diccionario
```

Codi 11: Comprovació supermodel

Un cop s'han processat totes les files del *DataFrame* inicial i el diccionari ja conté l'estructura jeràrquica amb tots els *distincts*, es fa una crida al model LLM per tal que busqui coincidències entre noms.

El *prompt* segueix l'estructura del cas anterior. Bàsicament, se li demana al model iterar entre totes les claus del diccionari i buscar coincidències. Com en tots els casos anteriors, es crea un *DataFrame* que fa de diccionari entre els noms 'bruts' i els noms 'nets'. En aquest punt hi ha dues possibles casuístiques pels models que han canviat de nom (els que no han canviat no s'ha de fer res més): (1) que el nom assignat sigui coincident amb un altre del diccionari o (2) que s'hagi netejat el nom, però que no coincideixi amb cap nom ja existent.

Per la primera opció, es crida a la funció `merge_nested_dicts`. Aquest codi ajunta les dues branques dels diccionaris amb coincidència de nom. És a dir, crea una sola branca amb tot el contingut de la branca 1 i de la branca 2 anteriors, evitant els duplicats en cas d'haver-n'hi.

Tot aquest procés es repeteix després per la capa de submodels de cada model i finalment per la capa de subsubmodel de cada submodel. Amb això, es té un diccionari 'net' i jeràrquic, sense duplicats dintre els seus valors. En aquest punt, ja es pot crear el mestre de models, l'únic necessari és una funció que converteixi el diccionari en un *DataFrame*. Aquest pas també serveix per 'xocar' la llista `ar` en la qual s'ha guardat prèviament tota la informació, contra aquest diccionari per netejar-ne el nom i tenir el mestre amb totes les variacions d'especificacions final.

La funció que s'encarrega d'aquest pas és `flatten_dict`, veure codi 12. Un requisit important que ha de complir aquesta funció és que ha de ser capaç de conservar la jerarquia i, per tant, guardar un registre per capa. Per exemple, si el diccionari fos:

```
diccionari = {
    "BMW": {"2 SERIES": {"218I": {"COUPE"}},
            "3 SERIES": {320D: {}}},
            }
```

El *DataFrame* esperat seria:

BRAND	***MODEL***
BMW	2 SERIES
BMW	2 SERIES 218I
BMW	2 SERIES 218I COUPE
BMW	3 SERIES
BMW	3 SERIES 320D

Taula 4: Representació del *DataFrame* conservant la jerarquia del diccionari a dalt mostrat

Això ha de ser així per tal de ser conseqüents amb la lògica aplicada per crear el diccionari on, si en algun moment s'ha creat el 218I com a submodel del 2 SERIES, és perquè una entrada del *dataset* original contenia només el valor 2 SERIES i, per tant, a l'hora de 'xocar-lo' necessitem tenir aquest valor controlat.

Aquesta funció explora recursivament cada nivell del diccionari. A mesura que baixa, va omplint una llista (path) amb les claus trobades. Quan ja ha identificat com a mínim dos nivells (per exemple, marca i model), guarda el camí. El resultat final és una llista de camins que després fàcilment amb l'ús de la llibreria `pandas` es converteix en *DataFrame*.

Un cop convertida la llista a *DataFrame* s'extreu ja el mestre de models. Ara només és necessari netejar els noms de la llista `ar` per acabar amb el mestre d'especificacions també.

Aquest contrast es fa amb la funció `contrast_entries_vs_master`. Aquesta funció fa una cerca semàntica amb un *threshold* de 0.9 i retorna la coincidència en model si n'hi ha o afegeix a la llista de no trobats per fer la posterior cerca en LLM per aquests casos.

Aquest cerca LLM segueix el mateix patró que en tots els casos anteriors. L'agent rep una llista amb els noms 'bruts' i el mestre on ha de cercar les coincidències i retorna una llista amb els noms 'nets'.

```python
def flatten_dict(data):
    result = []

    def recurse(current, path=None):
        if path is None:
            path = ['', '', '', '', '']

        if not current:
            return

        if path[0] and path[1]:
            result.append(path[:])

        for key, value in current.items():
            level = next((i for i, x in enumerate(path) if not x), None)
            if level is not None:
                new_path = path[:]
                new_path[level] = key
                result.append(new_path)
                recurse(value, new_path)

    recurse(data)
    return result
```

Codi 12: Funció `flatten_dict`

4.1.5 Diagrama de flux

Amb això ja teníem tot el procés finalitzat i havíem aconseguit generar els tres mestres: el de marques, el de models i el d'especificacions. A continuació, es presenta un diagrama de flux del script complet per tal de tenir una visió global del funcionament del codi.

Marques

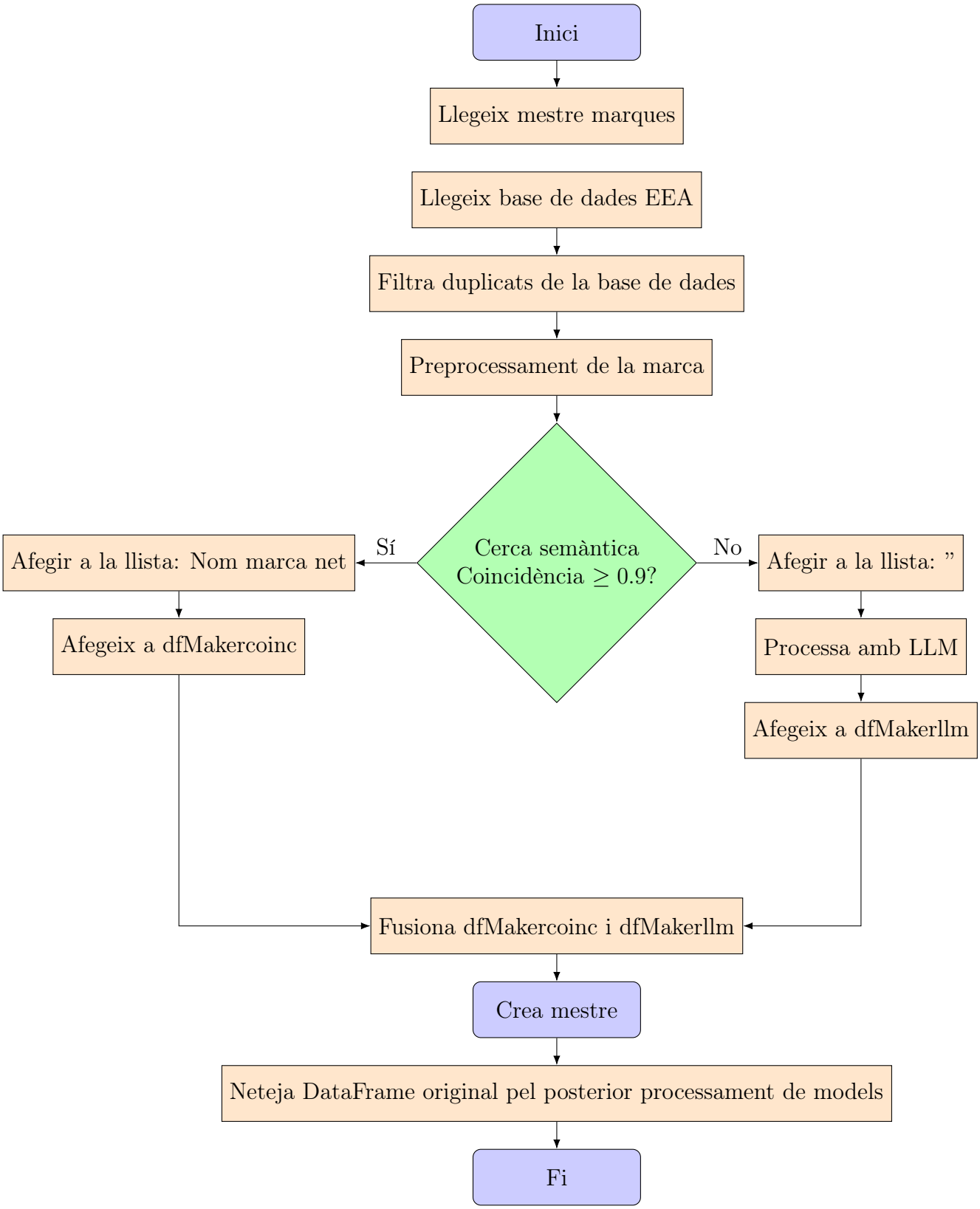


Figura 2: Diagrama de flux processament marques

Models

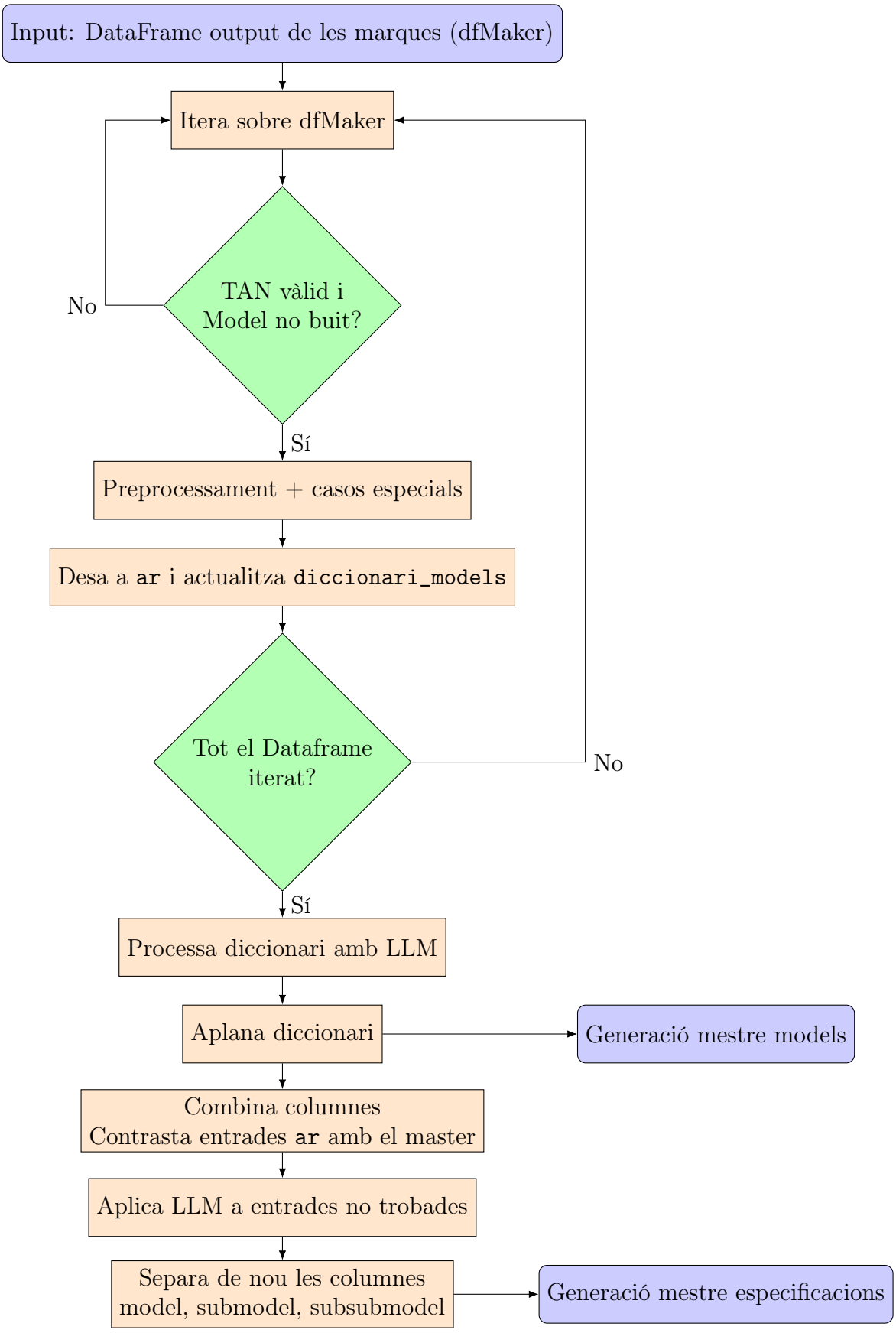


Figura 3: Diagrama de flux processament models

4.1.6 Revisió processament models i nou pipeline

Durant el desenvolupament inicial es va detectar que el procés implementat presentava certes redundàncies, especialment en etapes com la partició i reagrupació de models, així com en el contrast continuat amb el diccionari mestre. Aquest enfocament, si bé funcional, implicava una despesa computacional i temporal notable, amb múltiples iteracions que podien evitar-se mitjançant una reestructuració més eficient del flux de treball.

Des de feia temps, es contemplava la possibilitat d'optimitzar aquesta part del codi. Tanmateix, la prioritat inicial era garantir el funcionament correcte de totes les etapes. Un cop assolida aquesta estabilitat, es decidí passar a una segona fase de refinament, que implicava reformular la *pipeline* completa del procés de normalització i classificació de models.

Pipeline original

La versió inicial del *pipeline* seguia una estructura seqüencial on es realitzava:

- una primera passada per detectar models vàlids
- una generació inicial de l'estructura jeràrquica del diccionari
- una neteja dels noms sobre aquest diccionari, processant totes les capes d'aquest
- creació del *DataFrame* amb les columnes combinades i generació mestre models
- una comparació posterior contra aquest *DataFrame* semànticament
- una comparació posterior contra aquest *DataFrame* amb model LLM
- una separació de nou de les columnes en model, submodel i subsubmodel

Aquest procés suposava múltiples recorreguts sobre les mateixes dades, ús intensiu de memòria amb llistes temporals, i un sistema de correcció sobre el diccionari que incrementava el cost temporal.

Nova versió optimitzada

- La neteja i normalització dels models es fa en una sola passada durant la lectura del dataset, evitant passes addicionals.
- L'ús del LLM es concentra en un sol punt del procés, just després de la preparació de la llista de dades (`ar`), reduint la necessitat de correccions posteriors.
- S'elimina la necessitat de comparació contra un diccionari mestre previ, ja que aquest es genera a partir de les dades ja netejades.
- S'elimina el procés de combinar les columnes del *DataFrame*, ja que al 'netejar' directament sobre la llista `ar` aquesta ja conté la informació preparada.

Diagrama de flux de la nova *pipeline*

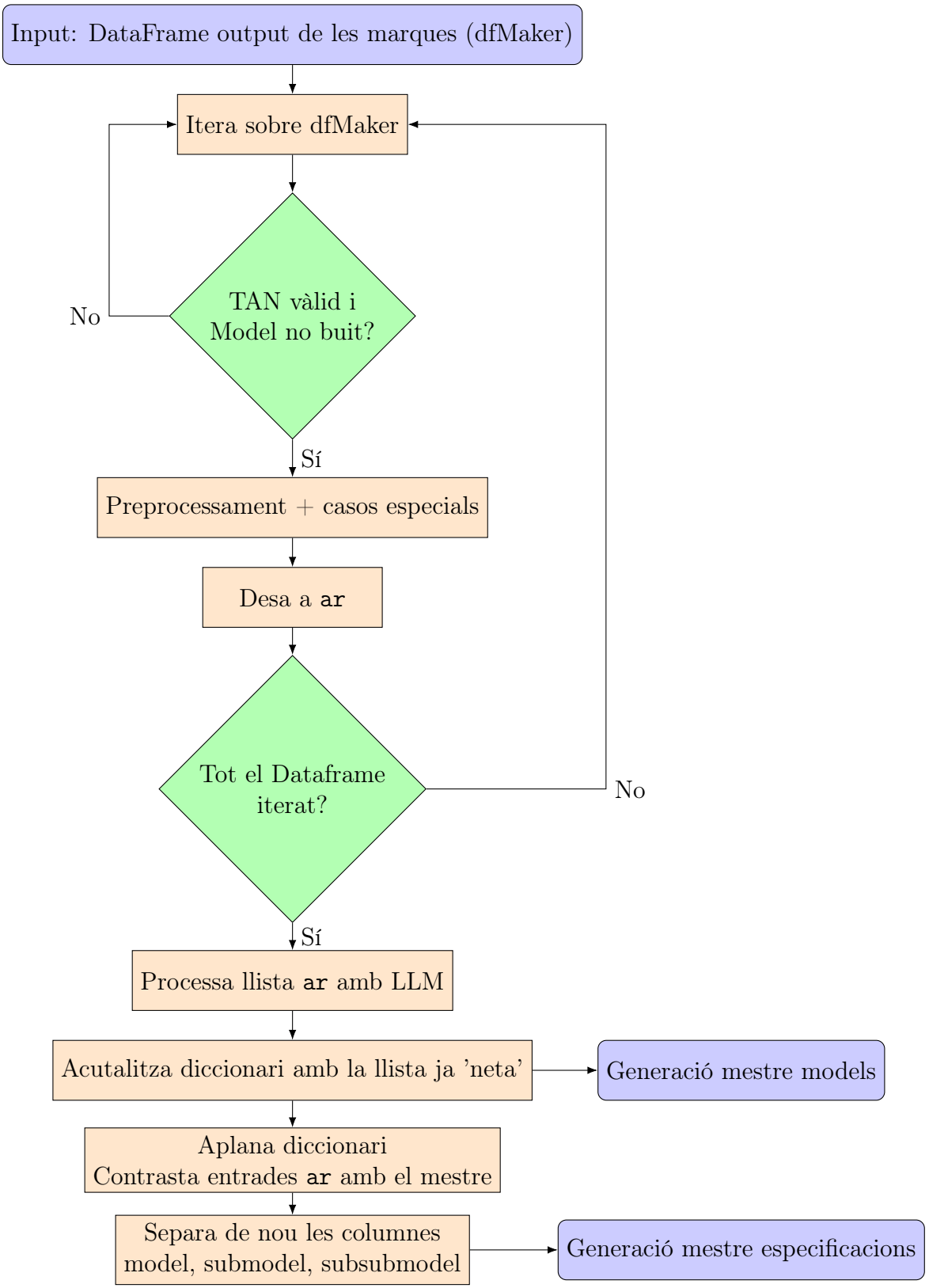


Figura 4: Diagrama de flux processament models amb la nova *pipeline*

Impacte temporal

A nivell computacional, aquest canvi ha comportat una reducció del temps d'execució estimada entre un 70% i un 80% (segons el volum de dades tractades), amb un menor ús de memòria i una millora substancial en la llegibilitat del codi. A la taula 5 podem veure les principals diferències en els *scripts* que han contribuït a aquesta considerable millora.

Aspecte	Versió Antiga	Nova Versió
Iteracions sobre dades	Fa múltiples passades sobre les dades	Fa només una passada principal
Temps de computació	Elevat, especialment per la duplicació de processaments i recorreguts	Reduït gràcies a la compactació de tasques i ús eficient de la memòria
Ús del LLM	El LLM es crida després de generar el diccionari, augmenta la feina	Es fa una única crida directa al LLM per netejar i normalitzar `ar`
Actualització del diccionari	Incremental dins del bucle	Agrupada i unificada després del LLM
Simplificació	Processos redundants i dependents	Flux clar i seqüenciat, amb ús de funcions optimitzades
Temps d'execució estimat	Més alt (mínim 2x recorreguts de dades, amb llistes temporals, comparacions, etc.)	Més baix, gràcies a reducció d'etapes i millor aprofitament del LLM
Claredat de la pipeline	Lògica trencada entre fases: neteja → diccionari → contrast → correcció	Lògica més clara

Taula 5: Comparativa entre l'script original i el nou pipeline optimitzat per models

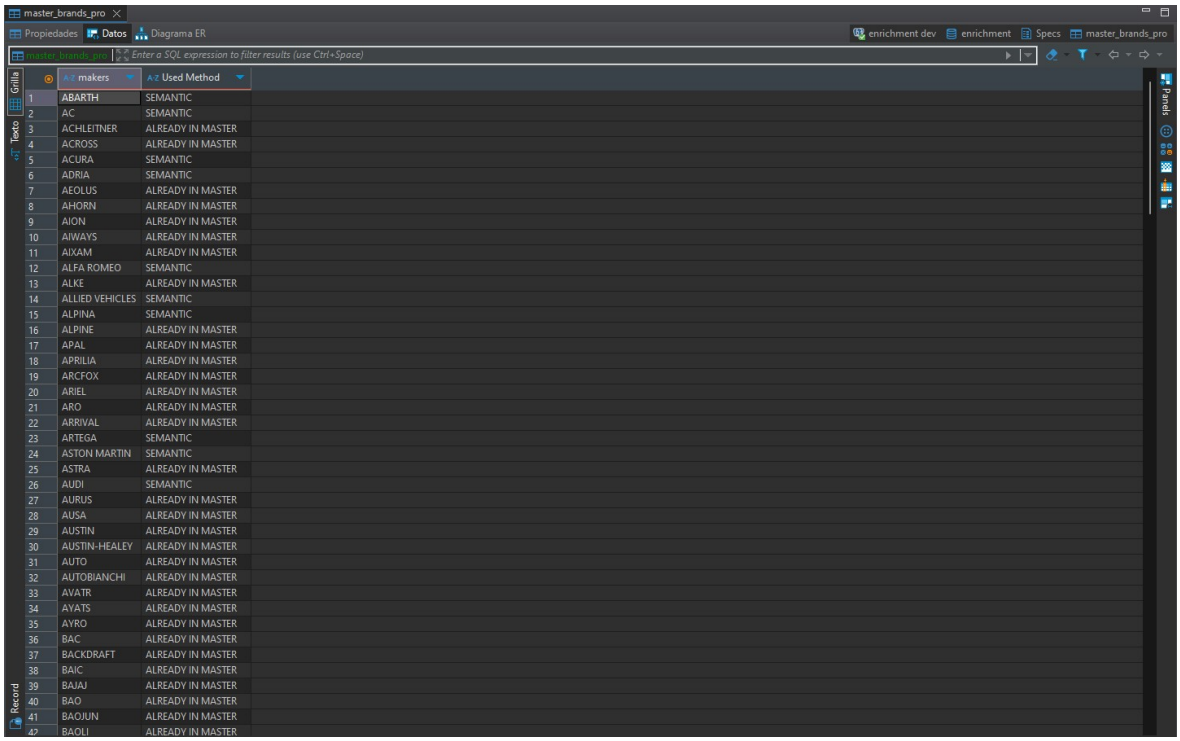
4.1.7 Emmagatzematge de les taules generades

Les tres taules generades: (1) mestre de marques, (2) mestre de models i (3) mestre d'especificacions, es pengen en una base de dades dins d'*Amazon Relational Database Service* (RDS). Així, s'aconsegueix preparar correctament el procés d'automatització descrit al punt 4.2, integrat completament dins els serveis d'*Amazon Web Services* (AWS), veure secció 3.5.

Per accedir i gestionar les dades emmagatzemades a RDS s'ha utilitzat DBeaver, veure secció 3.6, un *software* de gestió de bases de dades.

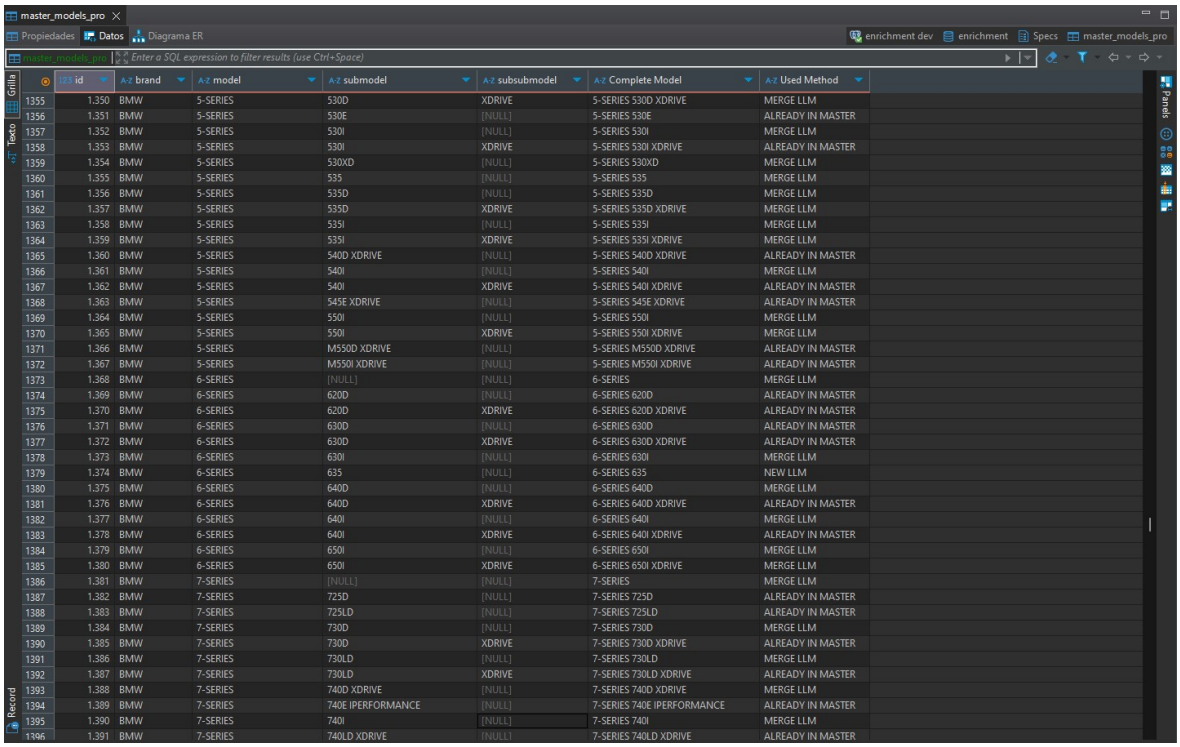
L'ús de DBeaver ha permès verificar la integritat de les dades carregades, realitzar proves de consultes complexes i validar el correcte funcionament de les taules generades al llarg del desenvolupament del projecte.

L'última de les columnes que es pot apreciar a les tres figures (figures 5, 6 i 7), la qual porta per nom '*Used Method*', és per fer un *trackeig* sobre com aquell registre ha arribat al diccionari i serà usada en la secció 5.1.



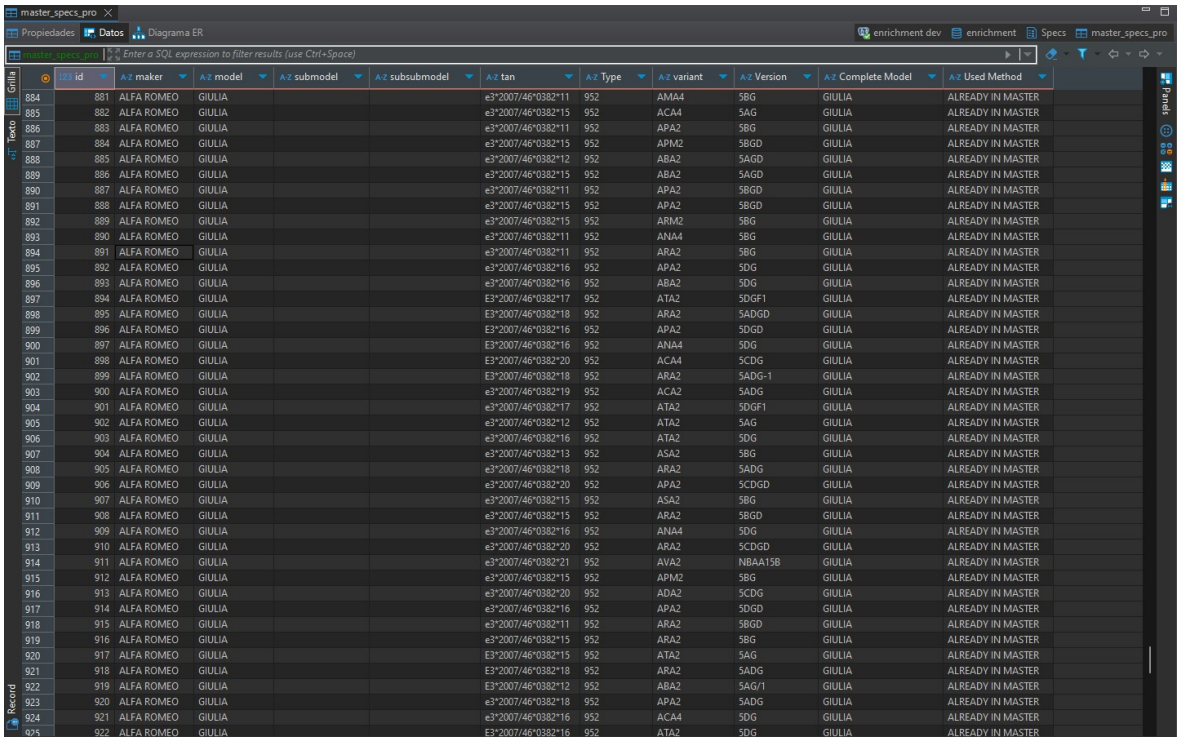
	makers	Used Method
1	ABARTH	SEMANTIC
2	AC	SEMANTIC
3	ACHLEITNER	ALREADY IN MASTER
4	ACROSS	ALREADY IN MASTER
5	ACURA	SEMANTIC
6	ADRIA	SEMANTIC
7	AEOLUS	ALREADY IN MASTER
8	AHORN	ALREADY IN MASTER
9	AION	ALREADY IN MASTER
10	AIWAYS	ALREADY IN MASTER
11	AIXAM	ALREADY IN MASTER
12	ALFA ROMEO	SEMANTIC
13	ALKE	ALREADY IN MASTER
14	ALLIED VEHICLES	SEMANTIC
15	ALPINA	SEMANTIC
16	ALPINE	ALREADY IN MASTER
17	APAL	ALREADY IN MASTER
18	APRILIA	ALREADY IN MASTER
19	ARCFOX	ALREADY IN MASTER
20	ARIEL	ALREADY IN MASTER
21	ARO	ALREADY IN MASTER
22	ARRIVAL	ALREADY IN MASTER
23	ARTEGA	SEMANTIC
24	ASTON MARTIN	SEMANTIC
25	ASTRA	ALREADY IN MASTER
26	AUDI	SEMANTIC
27	AURUS	ALREADY IN MASTER
28	AUSA	ALREADY IN MASTER
29	AUSTIN	ALREADY IN MASTER
30	AUSTIN-HEALEY	ALREADY IN MASTER
31	AUTO	ALREADY IN MASTER
32	AUTOBIANCHI	ALREADY IN MASTER
33	AVATR	ALREADY IN MASTER
34	AYATS	ALREADY IN MASTER
35	AYRO	ALREADY IN MASTER
36	BAC	ALREADY IN MASTER
37	BACKDRAFT	ALREADY IN MASTER
38	BAIC	ALREADY IN MASTER
39	BAJAJ	ALREADY IN MASTER
40	BAO	ALREADY IN MASTER
41	BAOJUN	ALREADY IN MASTER
42	BAOLI	ALREADY IN MASTER

Figura 5: Visualització mestre marques a DBeaver



	id	brand	model	submodel	subsubmodel	Complete Model	Used Method
1355	1.350	BMW	5-SERIES	530D	XDRIVE	5-SERIES 530D XDRIVE	MERGE LLM
1356	1.351	BMW	5-SERIES	530E	[NULL]	5-SERIES 530E	ALREADY IN MASTER
1357	1.352	BMW	5-SERIES	530I	[NULL]	5-SERIES 530I	MERGE LLM
1358	1.353	BMW	5-SERIES	530I	XDRIVE	5-SERIES 530I XDRIVE	ALREADY IN MASTER
1359	1.354	BMW	5-SERIES	530XD	[NULL]	5-SERIES 530XD	MERGE LLM
1360	1.355	BMW	5-SERIES	535	[NULL]	5-SERIES 535	MERGE LLM
1361	1.356	BMW	5-SERIES	535D	[NULL]	5-SERIES 535D	MERGE LLM
1362	1.357	BMW	5-SERIES	535D	XDRIVE	5-SERIES 535D XDRIVE	MERGE LLM
1363	1.358	BMW	5-SERIES	535I	[NULL]	5-SERIES 535I	MERGE LLM
1364	1.359	BMW	5-SERIES	535I	XDRIVE	5-SERIES 535I XDRIVE	MERGE LLM
1365	1.360	BMW	5-SERIES	540D XDRIVE	[NULL]	5-SERIES 540D XDRIVE	ALREADY IN MASTER
1366	1.361	BMW	5-SERIES	540I	[NULL]	5-SERIES 540I	MERGE LLM
1367	1.362	BMW	5-SERIES	540I	XDRIVE	5-SERIES 540I XDRIVE	ALREADY IN MASTER
1368	1.363	BMW	5-SERIES	545E XDRIVE	[NULL]	5-SERIES 545E XDRIVE	ALREADY IN MASTER
1369	1.364	BMW	5-SERIES	550I	[NULL]	5-SERIES 550I	MERGE LLM
1370	1.365	BMW	5-SERIES	550I	XDRIVE	5-SERIES 550I XDRIVE	MERGE LLM
1371	1.366	BMW	5-SERIES	M550D XDRIVE	[NULL]	5-SERIES M550D XDRIVE	ALREADY IN MASTER
1372	1.367	BMW	5-SERIES	M550I XDRIVE	[NULL]	5-SERIES M550I XDRIVE	ALREADY IN MASTER
1373	1.368	BMW	6-SERIES	[NULL]	[NULL]	6-SERIES	MERGE LLM
1374	1.369	BMW	6-SERIES	620D	[NULL]	6-SERIES 620D	ALREADY IN MASTER
1375	1.370	BMW	6-SERIES	620D	XDRIVE	6-SERIES 620D XDRIVE	ALREADY IN MASTER
1376	1.371	BMW	6-SERIES	630D	[NULL]	6-SERIES 630D	ALREADY IN MASTER
1377	1.372	BMW	6-SERIES	630D	XDRIVE	6-SERIES 630D XDRIVE	ALREADY IN MASTER
1378	1.373	BMW	6-SERIES	630I	[NULL]	6-SERIES 630I	MERGE LLM
1379	1.374	BMW	6-SERIES	635	[NULL]	6-SERIES 635	NEW LLM
1380	1.375	BMW	6-SERIES	640D	[NULL]	6-SERIES 640D	MERGE LLM
1381	1.376	BMW	6-SERIES	640D	XDRIVE	6-SERIES 640D XDRIVE	ALREADY IN MASTER
1382	1.377	BMW	6-SERIES	640I	[NULL]	6-SERIES 640I	MERGE LLM
1383	1.378	BMW	6-SERIES	640I	XDRIVE	6-SERIES 640I XDRIVE	ALREADY IN MASTER
1384	1.379	BMW	6-SERIES	650I	[NULL]	6-SERIES 650I	MERGE LLM
1385	1.380	BMW	6-SERIES	650I	XDRIVE	6-SERIES 650I XDRIVE	MERGE LLM
1386	1.381	BMW	7-SERIES	[NULL]	[NULL]	7-SERIES	MERGE LLM
1387	1.382	BMW	7-SERIES	725D	[NULL]	7-SERIES 725D	ALREADY IN MASTER
1388	1.383	BMW	7-SERIES	725LD	[NULL]	7-SERIES 725LD	ALREADY IN MASTER
1389	1.384	BMW	7-SERIES	730D	[NULL]	7-SERIES 730D	MERGE LLM
1390	1.385	BMW	7-SERIES	730D	XDRIVE	7-SERIES 730D XDRIVE	ALREADY IN MASTER
1391	1.386	BMW	7-SERIES	730LD	[NULL]	7-SERIES 730LD	MERGE LLM
1392	1.387	BMW	7-SERIES	730LD	XDRIVE	7-SERIES 730LD XDRIVE	ALREADY IN MASTER
1393	1.388	BMW	7-SERIES	740D XDRIVE	[NULL]	7-SERIES 740D XDRIVE	MERGE LLM
1394	1.389	BMW	7-SERIES	740E IPERFORMANCE	[NULL]	7-SERIES 740E IPERFORMANCE	ALREADY IN MASTER
1395	1.390	BMW	7-SERIES	740I	[NULL]	7-SERIES 740I	MERGE LLM
1396	1.391	BMW	7-SERIES	740LD XDRIVE	[NULL]	7-SERIES 740LD XDRIVE	ALREADY IN MASTER

Figura 6: Visualització mestre models a DBeaver



	id	maker	model	submodel	subsubmodel	tan	Type	variant	Version	Complete Model	Used Method
884	881	ALFA ROMEO	GIULIA			e3*2007/46*0382*11	952	AMA4	5BG	GIULIA	ALREADY IN MASTER
885	882	ALFA ROMEO	GIULIA			e3*2007/46*0382*15	952	ACA4	5AG	GIULIA	ALREADY IN MASTER
886	883	ALFA ROMEO	GIULIA			e3*2007/46*0382*11	952	APA2	5BG	GIULIA	ALREADY IN MASTER
887	884	ALFA ROMEO	GIULIA			e3*2007/46*0382*15	952	APM2	5BGD	GIULIA	ALREADY IN MASTER
888	885	ALFA ROMEO	GIULIA			e3*2007/46*0382*12	952	ABA2	5AGD	GIULIA	ALREADY IN MASTER
889	886	ALFA ROMEO	GIULIA			e3*2007/46*0382*15	952	ABA2	5AGD	GIULIA	ALREADY IN MASTER
890	887	ALFA ROMEO	GIULIA			e3*2007/46*0382*11	952	APA2	5BGD	GIULIA	ALREADY IN MASTER
891	888	ALFA ROMEO	GIULIA			e3*2007/46*0382*15	952	APA2	5BGD	GIULIA	ALREADY IN MASTER
892	889	ALFA ROMEO	GIULIA			e3*2007/46*0382*15	952	ARM2	5BG	GIULIA	ALREADY IN MASTER
893	890	ALFA ROMEO	GIULIA			e3*2007/46*0382*11	952	ANA4	5BG	GIULIA	ALREADY IN MASTER
894	891	ALFA ROMEO	GIULIA			e3*2007/46*0382*11	952	ARA2	5BG	GIULIA	ALREADY IN MASTER
895	892	ALFA ROMEO	GIULIA			e3*2007/46*0382*16	952	APA2	5DG	GIULIA	ALREADY IN MASTER
896	893	ALFA ROMEO	GIULIA			e3*2007/46*0382*16	952	ABA2	5DG	GIULIA	ALREADY IN MASTER
897	894	ALFA ROMEO	GIULIA			E3*2007/46*0382*17	952	ATA2	5DGF1	GIULIA	ALREADY IN MASTER
898	895	ALFA ROMEO	GIULIA			E3*2007/46*0382*18	952	ARA2	5ADGD	GIULIA	ALREADY IN MASTER
899	896	ALFA ROMEO	GIULIA			E3*2007/46*0382*16	952	APA2	5DGD	GIULIA	ALREADY IN MASTER
900	897	ALFA ROMEO	GIULIA			E3*2007/46*0382*16	952	ANA4	5DG	GIULIA	ALREADY IN MASTER
901	898	ALFA ROMEO	GIULIA			E3*2007/46*0382*20	952	ACA4	5CDG	GIULIA	ALREADY IN MASTER
902	899	ALFA ROMEO	GIULIA			E3*2007/46*0382*18	952	ARA2	5ADG-1	GIULIA	ALREADY IN MASTER
903	900	ALFA ROMEO	GIULIA			e3*2007/46*0382*19	952	ACA2	5ADG	GIULIA	ALREADY IN MASTER
904	901	ALFA ROMEO	GIULIA			e3*2007/46*0382*17	952	ATA2	5DGF1	GIULIA	ALREADY IN MASTER
905	902	ALFA ROMEO	GIULIA			e3*2007/46*0382*12	952	ATA2	5AG	GIULIA	ALREADY IN MASTER
906	903	ALFA ROMEO	GIULIA			e3*2007/46*0382*16	952	ATA2	5DG	GIULIA	ALREADY IN MASTER
907	904	ALFA ROMEO	GIULIA			e3*2007/46*0382*13	952	ASA2	5BG	GIULIA	ALREADY IN MASTER
908	905	ALFA ROMEO	GIULIA			e3*2007/46*0382*18	952	ARA2	5ADG	GIULIA	ALREADY IN MASTER
909	906	ALFA ROMEO	GIULIA			e3*2007/46*0382*20	952	APA2	5CDGD	GIULIA	ALREADY IN MASTER
910	907	ALFA ROMEO	GIULIA			e3*2007/46*0382*15	952	ASA2	5BG	GIULIA	ALREADY IN MASTER
911	908	ALFA ROMEO	GIULIA			e3*2007/46*0382*15	952	ARA2	5BGD	GIULIA	ALREADY IN MASTER
912	909	ALFA ROMEO	GIULIA			e3*2007/46*0382*16	952	ANA4	5DG	GIULIA	ALREADY IN MASTER
913	910	ALFA ROMEO	GIULIA			e3*2007/46*0382*20	952	ARA2	5CDGD	GIULIA	ALREADY IN MASTER
914	911	ALFA ROMEO	GIULIA			e3*2007/46*0382*21	952	AVA2	NBAA15B	GIULIA	ALREADY IN MASTER
915	912	ALFA ROMEO	GIULIA			e3*2007/46*0382*15	952	APM2	5BG	GIULIA	ALREADY IN MASTER
916	913	ALFA ROMEO	GIULIA			e3*2007/46*0382*20	952	ADA2	5CDG	GIULIA	ALREADY IN MASTER
917	914	ALFA ROMEO	GIULIA			e3*2007/46*0382*16	952	APA2	5DGD	GIULIA	ALREADY IN MASTER
918	915	ALFA ROMEO	GIULIA			e3*2007/46*0382*11	952	ARA2	5BGD	GIULIA	ALREADY IN MASTER
919	916	ALFA ROMEO	GIULIA			e3*2007/46*0382*15	952	ARA2	5BG	GIULIA	ALREADY IN MASTER
920	917	ALFA ROMEO	GIULIA			E3*2007/46*0382*15	952	ATA2	5AG	GIULIA	ALREADY IN MASTER
921	918	ALFA ROMEO	GIULIA			E3*2007/46*0382*18	952	ARA2	5ADG	GIULIA	ALREADY IN MASTER
922	919	ALFA ROMEO	GIULIA			E3*2007/46*0382*12	952	ABA2	5AG/1	GIULIA	ALREADY IN MASTER
923	920	ALFA ROMEO	GIULIA			e3*2007/46*0382*18	952	APA2	5ADG	GIULIA	ALREADY IN MASTER
924	921	ALFA ROMEO	GIULIA			e3*2007/46*0382*16	952	ACA4	5DG	GIULIA	ALREADY IN MASTER
925	922	ALFA ROMEO	GIULIA			E3*2007/46*0382*16	952	ATA2	5DG	GIULIA	ALREADY IN MASTER

Figura 7: Visualització mestre especificacions a DBeaver

4.2 Automatització del procés de generació i actualització del mestre

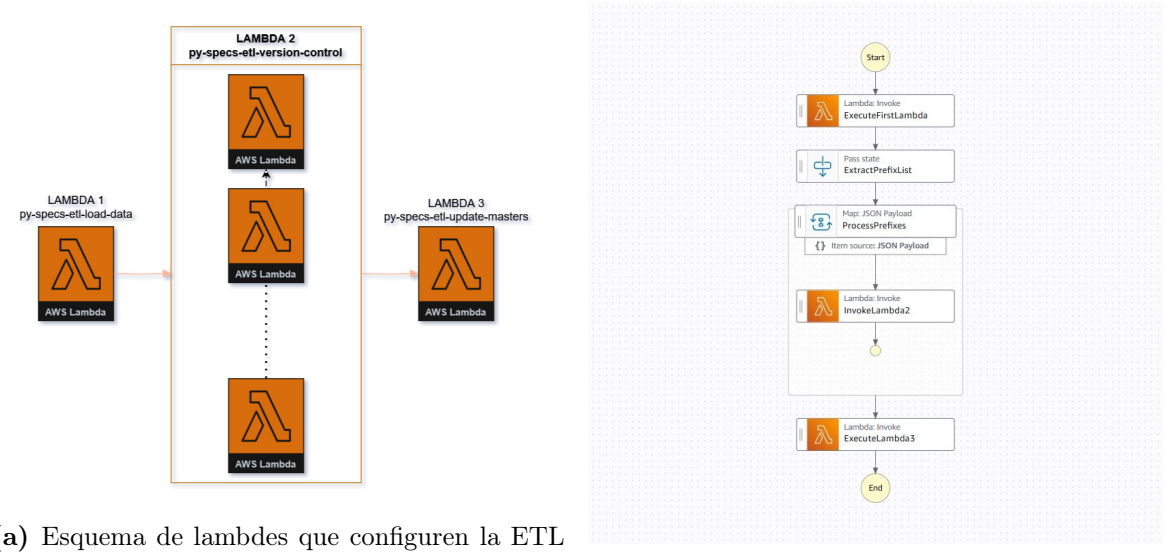
Un cop generada la primera versió del mestre, calia automatitzar el procés de la seva actualització amb l'objectiu de garantir la seva vigència i coherència al llarg del temps. Aquesta automatització permet incorporar nous registres de manera eficient, reduint l'esforç manual i assegurant que el mestre reflecteixi sempre el panorama actual de marques i models.

Sabent que la EEA publica noves dades cada quatre mesos, es va dissenyar un sistema que iniciés el procés d'actualització de forma autònoma seguint aquest mateix període.

Dins l'empresa és comú l'ús dels serveis integrats d'AWS i, per tant, es va decidir desenvolupar tota la *pipeline* d'ETL (*Extract, Transform, Load*) d'automatització en aquesta plataforma. L'elecció d'AWS permet aprofitar les seves capacitats escalables i la integració fluida entre serveis com *AWS Lambda*, *Amazon S3*, *AWS Step Funtions* o *Amazon CloudWatch*, els quals faciliten una gestió eficient de dades, la programació d'execucions periòdiques i el monitoratge dels processos.

El primer necessari era aconseguir aquestes noves dades. Aquesta feina, però, va ser proporcionada per un equip d'IDIADA INDIA, que és el que s'encarrega de fer *web-scraping*. Ells proporcionaven els nous fitxers CSV comprimits en ZIP en un *bucket* de S3.

4.2.1 State Machine



(a) Esquema de lambdes que configuren la ETL automàtica d'actualització del mestre

(b) Estructura de la *state machine*

Figura 8: *State machine*

Pel que fa a l'arquitectura, es va optar per construir un sistema de lambdes accionades i controlades per una màquina de *Step Functions*. L'estructura es pot veure a la figura 8a.

Durant el procés es farà ús de dos *buckets* d'AWS S3, un primer del qual es recolliran les dades proporcionades per l'equip d'India (`xplain-datalake-bronze`) i que es portaran al segon, un de propi en el qual es durà a terme tota la resta del procés (`specs-eea-raw-files`).

Aquest procés s'executa automàticament cada quatre mesos mitjançant la invocació de la primera funció lambda. La resta de funcions s'activen de manera seqüencial, garantint que cada etapa només s'iniciï un cop ha finalitzat l'anterior. En particular, per controlar l'execució paral·lela de les funcions Lambda 2 —una per cada fitxer detectat— s'utilitza un bloc de tipus *Map State*, el qual assegura que totes les instàncies de lambda 2 han finalitzat correctament abans de procedir amb la invocació de la tercera funció lambda. A la figura 8b es pot visualitzar l'estructura.

La configuració d'aquesta màquina d'estats es defineix mitjançant un fitxer en format JSON, on s'especifiquen les funcions Lambda que s'han d'invocar, els *payloads* que es transfereixen entre elles, així com els *buckets* d'origen i destinació i altres paràmetres necessaris per al correcte funcionament de l'entorn.

4.2.2 1a Lambda. Carregar les dades

La 1a lambda porta per nom *py-specs-etl-load-data* i, com el seu nom bé indica, s'encarrega de llegir les dades del *bucket* d'origen i portar-les al *bucket* propi.

El *bucket* d'origen presenta una estructura organitzada jeràrquicament, com es mostra a la figura 9. La ruta completa, d'ara endavant prefix, ('test_bucket/type=Cars/web-EEA/folder=web-scraping-versions/year=2025/month=3/') segueix un patró de partionament lògic que facilita l'accés i la gestió de les dades.

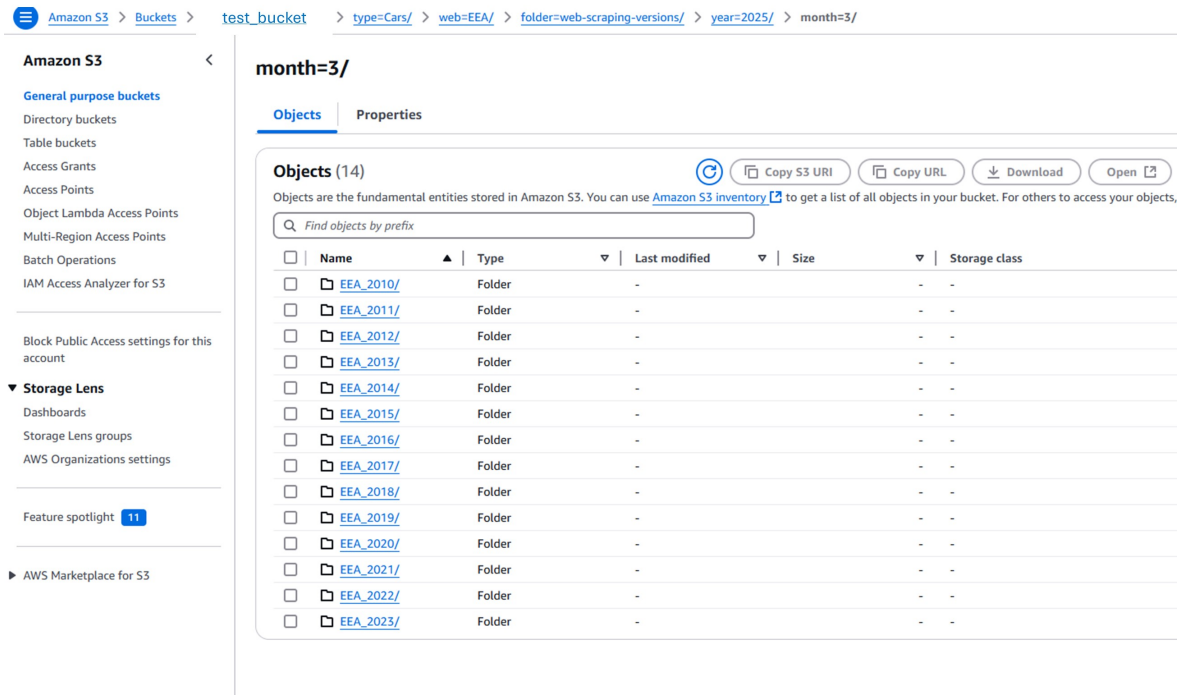


Figura 9: Estructura *bucket* d'origen

Com es pot observar a la barra de navegació superior, les últimes dues carpetes ('year=2025/' i 'month=3/') corresponen a particions temporals que indiquen l'any i el mes en què es va realitzar l'extracció i càrrega de les dades mitjançant el procés de *web-scraping*.

Dins d'aquesta partició mensual, trobem una estructura de diferents carpetes, cadascuna corresponent a un any específic de dades recopilades. Cada una d'aquestes carpetes conté un arxiu comprimit en format ZIP amb tota la informació extreta per a aquell any en particular en CSVs.

La lambda s'encarrega, entre d'altres, de mantenir aquesta estructura en el *bucket* de destí. Per a fer-ho, en el *bucket* propi es guarda un fitxer TXT amb la informació de l'últim any i mes que s'ha llegit. Amb això, i sabent que l'execució és cada 4 mesos, es calcula el nou prefix on la lambda ha de llegir la informació.

En aquest moment, la lambda va carpeta per carpeta i per cada ZIP que hi troba, l'obre i puja al *bucket* de destí tots els CSV que hi ha dins.

Finalment, es puja a cada carpeta 'anual' del *bucket* de destí (p. ex., 'EEA_2010/', 'EEA_2011/', etc.) el que es coneix com un arxiu *manifest*. Aquest arxiu, generalment en format JSON o YAML, no només indica que el procés d'extracció i càrrega ha finalitzat correctament per a aquell any específic, sinó que també pot contenir metadades sobre els fitxers processats, incloent-hi rutes, timestamps i estatus de validació.

Quan es detecta la creació d'aquest arxiu, la *state machine*, implementada mitjançant *AWS Step Functions*, inicia de forma asíncrona una 'lambda 2', encarregada de processar les dades d'aquell any concret, per cada *manifest* detectat.

En el cas de la figura 9, on es poden observar 14 carpetes anuals, el sistema activaria paral·lelament 14 'lambda 2'. Aquesta execució paral·lela permet optimitzar significativament el temps total de processament, ja que cada funció Lambda treballa de manera independent amb les dades d'un any específic. A més, tal com es veurà més endavant, és indispensable pel correcte funcionament de la ETL que la totalitat de les 'lambda 2' hagin finalitzat per iniciar la 3a i última lambda.

Script

A *Python*, la llibreria que permet interactuar amb els serveis d'AWS és *boto3*, veure secció 3.1. Aquesta llibreria proporciona una interfície senzilla i coherent per accedir a múltiples serveis d'AWS. Dins el codi desenvolupat, es crea un *client* per a cada servei específic que es fa servir, com S3, per exemple.

El primer que fa la lambda és cridar la funció `get_next_prefix`. Aquesta funció és l'encarregada d'obtenir el mes i l'any de l'última lectura de dades per tal de construir el prefix on llegir les dades noves. El seu funcionament és molt simple, primer de tot intenta llegir l'últim prefix del fitxer TXT guardat al *bucket*. Si existeix, el parteix, llegeix mes i any i suma 4 mesos a la data per crear el nou prefix. Si el fitxer no existeix, ja que pot ser la 1a execució, mitjançant la data actual l'entra al cicle d'actualitzacions, cada quatre mesos (març, juliol, novembre).

Tot seguit, s'obté una llista on cada element és una de les carpetes anuals. Al següent pas s'itera sobre aquesta llista i mitjançant un directori temporal de la llibreria `tempfile` es descarrega el ZIP que conté cada carpeta, tot fent ús de la llibreria `zipfile` i les seves funcions integrades. Cadascun dels CSV continguts al ZIP són pujats al *bucket* de destí, tot mantenint el prefix i, per tant, mantenint l'estructura del *bucket* d'origen.

Finalment, es crida la funció `save_processed_prefix`, que té per objectiu actualitzar el fitxer `.txt` amb el prefix que s'ha processat. A més, es generen i pengen els fitxers *manifest* en format JSON a cadascuna de les carpetes del *bucket* de destinació.

4.2.3 2a Lambda. Control de versions

Aquesta funció lambda s'encarrega de detectar canvis en fitxers CSV emmagatzemats al *bucket*. En concret, identifica files afegides o modificades respecte a un conjunt de dades de referència, que correspon als fitxers de l'anterior actualització. L'objectiu d'això és disminuir el nombre de crides a fer al model LLM en la 3a lambda, que s'encarregarà d'actualitzar el mestre.

Així, no caldrà processar tot el fitxer de nou, sinó aquells registres modificats o algun d'afegit. Només caldrà processar un fitxer sencer en cas que la EEA publiqui les dades d'un any més recent i que encara no havia publicat.

La lambda crea un fitxer de referència amb tots els fitxers que es troben en la mateixa carpeta anual de l'execució anterior. Per exemple, per la lambda que correspon a l'any 2015 (és important recordar que s'activen diverses instàncies d'aquesta funció en paral·lel, una per cada carpeta anual que existeixi), si l'última execució va ser en data 11/2024, aquesta s'està duent a terme en data 03/2025. Llavors aquesta lambda anirà al prefix corresponent a l'execució anterior i la carpeta anual del 2015 (type=Cars/web-EEA/folder=web-scraping-versions/year=2024/month=11/EEA_2015‘) i crearà el fitxer de referència amb tots els CSV que hi troba dins (concatenant-los).

Tot seguit cerca els registres modificats i afegits i penja, en una carpeta diferent, un CSV amb aquests. Finalment, elimina tots els fitxers que hi havia en el prefix de l'última execució, ja que, a la pròxima, aquesta execució serà la que es farà servir de referència. Així, s'aconsegueix minimitzar la memòria d'emmagatzematge usada.

Script

Mitjançant la funció `get_previous_prefix` la funció calcula el prefix on cercar els fitxers de referència. Aquesta funció funciona igual a la ja explicada `get_next_prefix`. Amb aquesta informació, es crida la funció `create_reference_file`, encarregada de crear el fitxer de referència. Aquesta funció cerca al prefix calculat fitxers CSV, si no en troba (a la 1a execució, per exemple) no retorna cap fitxer i si en troba els descarrega en un directori temporal i crea el fitxer de referència tot concatenant-los.

Si aquesta funció no retorna cap fitxer de referència, aquesta lambda acaba aquí i la 3a lambda haurà d'executar-se amb tot el conjunt de dades. Si retorna un fitxer de referència, la funció continua cridant la funció `compare_csv_files`. Aquesta funció rep com a paràmetres una llista amb els *paths* a tots els fitxers CSV nous i el *path* al directori temporal on s'allotja el fitxer de referència.

Primer de tot, llegeix per *chunks* el fitxer de referència i el guarda en un *DataFrame*. Tot seguit itera sobre cada CSV amb les dades noves. Per cada iteració, el procés és el següent:

1. Mitjançant l'ús de llistes tipus *set*, es busquen *id*'s, i per tant registres, afegits i comuns. Aquests són els dos casos que es volen tractar. Veure codi 13. Els afegits ja estaran identificats i pels comuns caldrà revisar si han estat modificats o no. El *DataFrame* `ref_df` conté tota la informació passada que es farà servir de referència, mentre que `new_df` conté les dades del primer CSV del procés nou d'actualització.

```
ref_ids = set(int(id_) for id_ in ref_df[id_column].fillna(0))
new_ids = set(int(id_) for id_ in new_df[id_column].fillna(0))

# Calculate ID differences
added_ids = new_ids - ref_ids
common_ids = ref_ids.intersection(new_ids)
```

Codi 13: Ús de *sets* per cercar registres afegits i comuns

2. En un *DataFrame* anomenat `added_rows_df`, s'afegeixen tots els registres dels *id*'s afegits.

3. Pels *id*'s comuns, el tractament no pot ser tan directe, sinó que s'ha de comprovar si el registre ha estat modificat o no. Primer de tot, s'extreu la part de `ref_df` i de `new_df` que contenen els *id*'s comuns. Llavors, s'uneixen els dos conjunts en un *DataFrame* anomenat `merged` utilitzant *id* com a clau i afegint sufixos per diferenciar les columnes de referència (`_ref`) i les noves (`_new`). Per a cada columna, excepte `ID`, comprova si hi ha diferències entre el valor antic i el nou. Considera diferències tant si els valors són diferents (després d'eliminar espais i convertir a cadena), com si un d'ells és `NaN` i l'altre no, però ignora els casos en què ambdós valors són buits. Després, es construeix un *DataFrame* `modified_rows_df`. Veure el codi 14.

Quan s'acaben totes les iteracions (s'han llegit i tractat tots els CSV nous), es crea un *DataFrame* `versioned_df` tot concatenant tots els anteriors amb registres afegits i modificats. Aquest *DataFrame* representa la versió actualitzada dels canvis respecte al període anterior i és el resultat final retornat per la funció de comparació `compare_csv_files`.

El següent fragment de codi s'encarrega de desar les dades en format CSV al *bucket* d'S3. Si la variable `versioneddf` no és `None` (s'han trobat files afegides i/o modificades), primer crea un *buffer*[13] en memòria amb `io.StringIO()`, hi escriu el *DataFrame* `versioneddf` i després puja aquest contingut al *bucket* especificat.

[13]Un *buffer* és un espai temporal de memòria utilitzat per emmagatzemar dades de forma transitòria mentre s'estan transferint d'un lloc a un altre. Evita escriure a disc i és molt útil per *scripts* temporals, ideal per treballar amb APIs com *boto3*.

```
ref_subset = ref_df[ref_df[id_column].isin(common_ids)]
new_subset = new_df[new_df[id_column].isin(common_ids)]

# Merge and compare columns
merged = ref_subset.merge(new_subset, on='ID', suffixes=('_ref', '_new'))
modified_mask = False
for col in specs_cols:
    if col != 'ID':
        null_different = merged[f'{col}_ref'].isna() !=
        ↪  merged[f'{col}_new'].isna()

        col_ref = merged[f'{col}_ref'].astype(str).str.strip()
        col_new = merged[f'{col}_new'].astype(str).str.strip()

        values_different = col_ref != col_new
        both_empty = (col_ref == '') & (col_new == '')
        col_modified = ((values_different & ~both_empty) | null_different)
        modified_mask |= col_modified

modified_rows_df =
↪  new_subset[new_subset['ID'].isin(merged[modified_mask]['ID'])]
```

Codi 14: Tractament de *id*'s comuns

Finalment, es crida la funció `delete_reference_files_and_prefix` que elimina els fitxers i prefix d'on s'han llegit les dades usades de referència ja que, en futures iteracions, ja no faran falta perquè les actuals seran les de referència. Així, s'aconsegueix mantenir el *bucket* en un estat òptim de memòria, no generant sobrecostos per l'excés d'emmagatzematge usat.

4.2.4 3a Lambda. Actualització dels mestres

Aquesta última lambda és l'encarregada d'executar l'actualització dels mestres. Per fer-ho, combina els codis ja explicats anteriorment en el punt 4.1 usats per generar-los des de zero.

La lambda comença cercant al prefix on s'han guardat tots els `versioneddf` tots aquests fitxers CSV. El prefix el rep com a paràmetre gràcies a l'ús del servei *AWS Step Functions* que permet la transferència d'informació entre les lambdes dins d'un mateix flux d'execució. Per aquest motiu, és necessari que totes les lambdes 2 hagin acabat abans d'executar aquesta tercera, ja que, si no, no es disposaria de la informació completa.

A més, si per cada lambda 2 s'executés una lambda 3, podria passar que els diferents `versioneddf` generats per cada lambda 2 continguessin informació repetida, el que podria portar problemes a l'hora d'actualitzar els mestres.

Un altre cas problemàtic es presentaria si una funció lambda 2 finalitzés i, immediatament, s'invoqués la seva corresponent lambda 3, aquesta llegiria l'estat actual dels mestres i iniciaria el procés d'actualització. Si poc després una altra lambda 2 finalitzés i desencadenés la seva lambda 3, aquesta també llegiria els mestres en el mateix estat anterior i sobreescriuria els canvis aplicats per l'anterior. En conseqüència, només es conservarien els resultats de la darrera execució, perdent-se les modificacions aplicades prèviament.

Per evitar aquestes condicions de cursa i garantir la coherència de les dades, és fonamental que la consolidació dels resultats es realitzi en una única execució un cop finalitzades totes les funcions Lambda 2.

Un cop ha trobat tots els fitxers amb les dades de les versions, llegeix també els mestres actuals. En aquest punt executa les funcions d'actualització. Per les marques, el procés és idèntic a l'anterior: mitjançant l'algorisme semàntic busca coincidències, pels que no en troba en cerca de nou amb el model LLM i, finalment, per les marques que encara no s'ha trobat relació al mestre, un nou agent LLM considera si es tracta d'una marca nova (no encara al mestre) o d'un registre mal entrat.

La idea és la mateixa pels models, però, per motius ja explicats, sense algorisme semàntic: buscar relacions i, si no en troba, considerar-lo, o no, com un model nou.

Finalment, es penja aquesta nova versió dels mestres a la base de dades de RDS i s'eliminen els arxius `versioneddf` del *bucket*.

Script

El *script* comença llegint tots els CSV que contenen la informació versionada i també els mestres actuals. Per convertir el mestre de marques en un diccionari, es fa servir la funció `dataframe_to_dict` mostrada al codi 15.

```python
def dataframe_to_dict(df):
    result = {}
    for _, row in df.iterrows():
        levels = [str(val).strip() for val in row[1:-1] if val]

        if not levels or len(levels) < 2:
            continue

        current = result
        for i, level in enumerate(levels[:-1]):
            if level not in current:
                current[level] = {}
            current = current[level]

        if levels[-1] not in current:
            current[levels[-1]] = {}
    return result
```

Codi 15: Funció `dataframe_to_dict`

Per a cada fila, extreu els valors de la segona fins a la penúltima columna (la primera conté la marca i l'última el mètode usat per guardar-lo al mestre), ignorant-los si són buits, i els considera com una seqüència de nivells d'una jerarquia. Si la fila no té almenys dos nivells, s'omet. A continuació, la funció construeix progressivament un diccionari, afegint cada nivell com una clau dins del nivell anterior. El resultat final és un diccionari que representa l'estructura jeràrquica definida per les columnes del *DataFrame*.

Tot seguit, es filtren les dades del *DataFrame* que conté tots els `versioneddf` concatenats, ja que, com bé s'ha explicat, com que cada lambda 2 en genera un aquests poden contenir informació duplicada.

Les funcions que processen les marques i els models són idèntiques a les ja explicades al punt 4.1, i per això no s'explicaran de nou. Un cop s'han processat, el resultat és un *DataFrame*, un per cada mestre realment, que combina la informació que ja hi havia al mestre més marques i models afegits en aquesta iteració. Per tant, per guardar aquesta nova versió dels mestres només és necessari esborrar l'anterior i penjar-hi aquesta.

Finalment, s'eliminen tots els arxius `versioneddf`, alliberant espai en el *bucket* per futures iteracions.

5 Validació dels resultats

5.1 Cas d'ús: Neteja de la base de dades de Pampol Samples

Amb el propòsit d'avaluar l'eficàcia dels mestres treballats, es va decidir fer un test tot netejant una de les bases internes de l'empresa.

Es va optar per tractar de netejar l'actual base de dades de Pampol Samples. Aquesta decisió s'alinea amb el procés de transició de la gestió de dades de mostres de Pampol Samples a SPECS, tal com s'ha detallat al punt 2.1, proporcionant així un cas d'ús real amb valor afegit per l'organització.

Aquesta base de dades inclou informació de totes les mostres que entren a l'empresa. Per tant, era necessari filtrar, mitjançant una *query*, i obtenir el subconjunt de dades que interessa. Cal destacar que aquesta base de dades presenta una arquitectura relacional complexa, estructurada en diverses taules interconnectades mitjançant *id*'s. Les taules que inclouen la informació que es necessita són:

- `[SAMPLE]`: conté informació bàsica de la mostra: descripció, client, **categoria**...
- `[SAMPLE_SAMPLE_SPECIFICATION]`: conté la informació dels camps **marca** i **model** (en el cas que la mostra sigui un vehicle).

Aquestes dues bases es relacionen a través d'un índex que a la primera de les taules apareix sota el nom `SMP_ID` i a la segona `SSS_SAMPLE_ID`.

La *query* que relaciona aquestes bases de dades és:

```
1  SELECT
2      s.[SMP_ID],
3      s.[SMP_DESCRIPTION],
4      s.[SMP_CATEGORY_ID],
5      s.[SMP_CLIENT_NAME],
6      sss.[SSS_SAMPLE_SPECIFICATION_ID] s_spec_id,
7      sss.[SSS_NUMERIC_VALUE],
8      sss.[SSS_CHAR_VALUE],
9      sss.[SSS_LIST_ID],
10     sss.[SSS_BOOLEAN_VALUE]
11 FROM [APP_PAMPOL_SAMPLES].[dbo].[SAMPLE] s
12 LEFT JOIN [APP_PAMPOL_SAMPLES].[dbo].[SAMPLE_SAMPLE_SPECIFICATION] sss
13     ON sss.SSS_SAMPLE_ID = s.SMP_ID
14 WHERE s.[SMP_CATEGORY_ID] in (85, 87, 88, 96, 97, 103, 104, 110, 112, 114,
   ↪  115)
15 ORDER BY s.[SMP_ID] ASC, s_spec_id DESC
```

Codi 16: *Query* a la base de dades de Pampol Samples

A la línia 14 de la query 16 és on es filtra per tal d'obtenir només les mostres categoritzades com a vehicles (les categories separen els vehicles en tipus: M1, N1, etc., i també per si són gasolina, dièsel, híbrids o elèctrics).

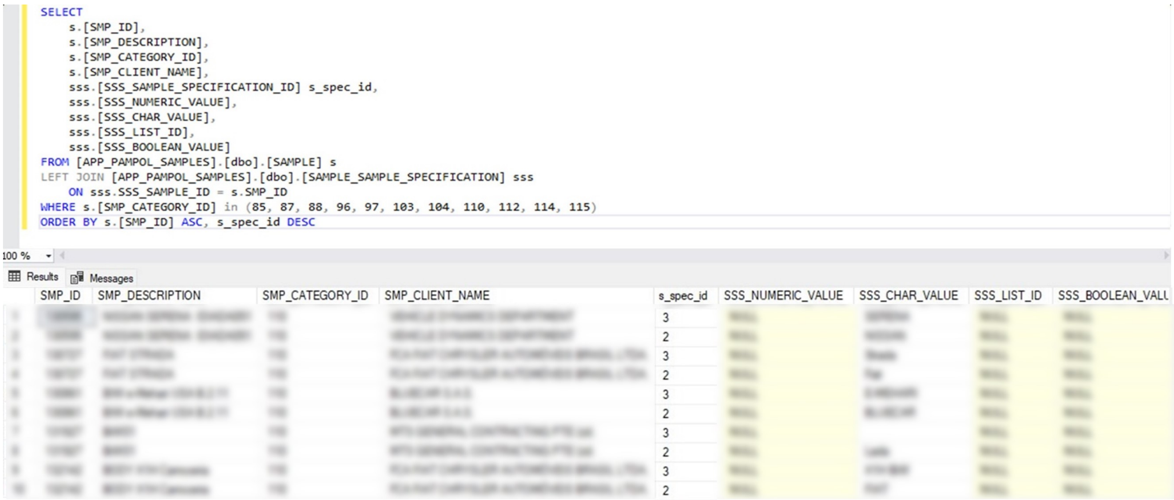


Figura 10: Visualització *query* i taula obtinguda

Com s'aprecia a la figura 10, `s_spec_id` és un atribut amb valors numèrics. De la mateixa base de dades es llegeix una taula amb les relacions entre el valor numèric i el nom descriptiu que aquest representa. És aquest atribut el que indica quina informació aporta el registre: 2 és marca i 3 és model.

5.1.1 Script de neteja

Per tal de fer la neteja, s'implementa un codi en *python*. Comença llegint aquestes dades i guardant-les en un *DataFrame*. El primer que fa es trobar quin percentatge de mostres (l'*id* de la mostra ve representat per `SMP_ID`) té emplenats els camps marca (`s_specs_id` = '2') i model (`s_specs_id` = '3'). Descarta tots els que no tenen nom designat. Pels quals si hi ha la informació, comença el procés de neteja. Aproximadament, un 65% de les entrades no inclou cap informació en cap dels dos camps, mentre que el 35% restant sí que té dades.

La neteja segueix la següent estratègia: per cada marca i model es busca la seva coincidència als mestres. Es pot trobar l'anàlisi dels resultats al punt 5.1.2. El primer pas és el de netejar les marques. Com bé s'ha explicat anteriorment, la menor variabilitat en la forma d'escriure una marca permet fer un preprocessament semàntic. Així, amb un $threshold = 0.9$ es busquen coincidències al mestre.

Pels casos en què no es troba coincidència, es fa una crida al model LLM per tal que busqui relacions. L'estructura del *prompt* és idèntica a la que ja s'ha descrit amb anterioritat. L'únic que el fa destacable és una breu descripció de com tractar el cas en què com a marca es rep un nom d'un grup fabricant, com per exemple *PSA Groupe* (Peugeot, Citroën, DS, Opel i Vauxhall).

Aquest cop, l'agent LLM no només rebrà la llista de noms de marques 'bruts', sinó que també una de models de cada marca. Així, si per un grup fabricant s'identifica que els models rebuts només correspon a una de les marques que agrupa, se li designa aquest nom.

Amb tot aquest procés, s'obté un *DataFrame* amb: Marca 'bruta' ('`Dirty Maker`'), marca 'neta' ('`Clean Maker`') i el mètode usat ('`Used Method`'), que pot ser '`SEMANTIC`', '`LLM`' o bé '`NOT CONSIDERED`' en cas que no es trobi cap relació al mestre.

Amb aquest, es va al *DataFrame* original i es modifiquen els noms de les marques pels noms 'nets' i es descarten els que no s'han relacionat.

Pels models es comença també fent una cerca semàntica, però aquest cop amb un *threshold* = 1, és a dir, la coincidència ha de ser exacte. Així, s'aconsegueix disminuir el nombre de crides al model LLM sense fer relacions incorrectes.

Tot seguit, s'agrupen els models que encara no han estat relacionats en *batches* de la mateixa marca i se li passen a l'agent perquè busqui coincidències amb models del mestre.

Una de les modificacions més importants en aquest cas és la instrucció de no suprimir informació. Si l'agent rep com a model 'brut' A3 COUPE i la millor coincidència trobada al mestre és A3, no se li assigna aquest nom. Realment, aquesta decisió dependrà de la base de dades que s'estigui netejant, del propòsit final i del grau de precisió esperat. Com que per aquest projecte la neteja s'està fent per tal de validar els mestres, s'espera la màxima precisió i, per tant, aquest cas es considera com que no s'ha trobat cap *match*.

Com en el cas de les marques, s'obté un *DataFrame* final, però aquest cop de quatre columnes: Marca(`Brand`), model 'brut' ('`Dirty Model`'), model 'net' ('`Clean Model`') i el mètode usat ('`Used Method`'), que, de nou, pot ser '`SEMANTIC`', '`LLM`' o bé '`NOT CONSIDERED`' en cas que no es trobi cap relació al mestre.

5.1.2 Anàlisi dels resultats

L'anàlisi dels resultats de la neteja, que permetrà extreure sòlides conclusions sobre l'eficàcia de les taules mestres, es va fer mitjançant la implementació d'un *dashboard*[14] a Qlik, veure punt 3.7.

Abans d'iniciar l'anàlisi cal destacar que la qualitat dels resultats de la neteja depèn no només dels mestres generats en aquest projecte, sinó també de la base de dades que es busca netejar. En aquest cas, es trobaran molts casos problemàtics: el camp marca i model sense emplenar, mostres mal categoritzades que realment no corresponen a vehicles i registres on els camps no contindran el nom comercial sinó codis interns que fan servir els tècnics de l'empresa.

Pel fet que busquem valorar l'eficiència dels mestres, s'espera que la neteja descarti tots aquests casos, encara que, evidentment, depenent de la finalitat real de la neteja, hi podria haver variacions, com per exemple les dades mal categoritzades identificar-les i recategoritzar-les.

Descripció del *dashboard*

El *dashboard* inclou 3 fulls: 1 pel control de la neteja de marques, (1) `PAMPOL BRANDS CLEANING`, i 2 pel control de la neteja de models, (2) `PAMPOL MODELS CLEANING` i (3) 2. `PAMPOL MODELS CLEANING`.

[14] Un *dashboard* és un tipus d'interfície gràfica d'usuari que proporciona vistes ràpides de les dades rellevants per a un objectiu o procés concret mitjançant una combinació de visualitzacions i informació resumida.

Pampol brands cleaning

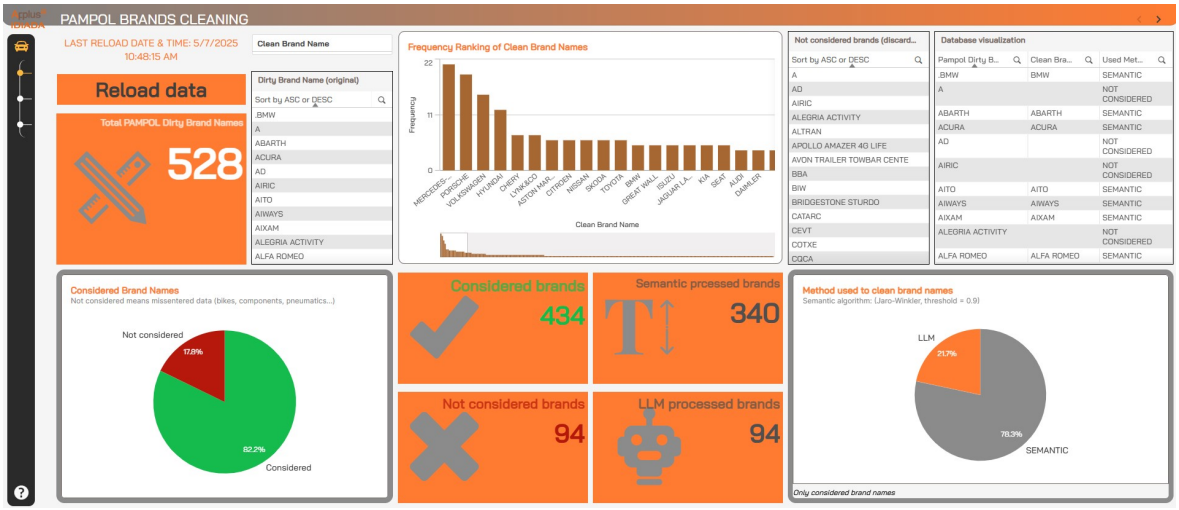


Figura 11: Full del *dashboard* neteja marques. Full 1

Com s'aprecia a la figura 11, la informació que es mostra en aquest cas és, d'esquerra a dreta i de dalt a baix:

- Mostra el títol `PAMPOL BRANDS CLEANING` i la data i hora de l'última actualització de les dades (5/7/2025 10:48:15 AM), en format (M/D/A).
- Nombre total de noms de marques 'bruts' rebuts (528).
- Llistat de marques 'brutes' amb un botó de filtratge. Permet seleccionar una marca i veure quines marques 'brutes' se li han atribuït.
- Diagrama de barres que representa la distribució de freqüència de les marques després del procés de neteja. L'eix vertical indica el nombre d'aparicions i l'eix horitzontal mostra les marques normalitzades.
- Llistat de marques no considerades.
- Visualització de la base de dades que s'està consultant.
- Gràfic circular per mostrar el percentatge de marques considerades (82,2%) i no considerades (17,8%).
- Quatre indicadors visuals que quantifiquen:
  - Marques considerades: 434 registres
  - Marques processades mitjançant algorisme semàntic: 340 registres
  - Marques no considerades: 94 registres
  - Marques processades mitjançant LLM: 94 registres
- Gràfic circular per mostrar el percentatge de marques considerades netejades amb algorisme semàntic (78,3%) i LLM (21,7%).

Pampol models cleaning

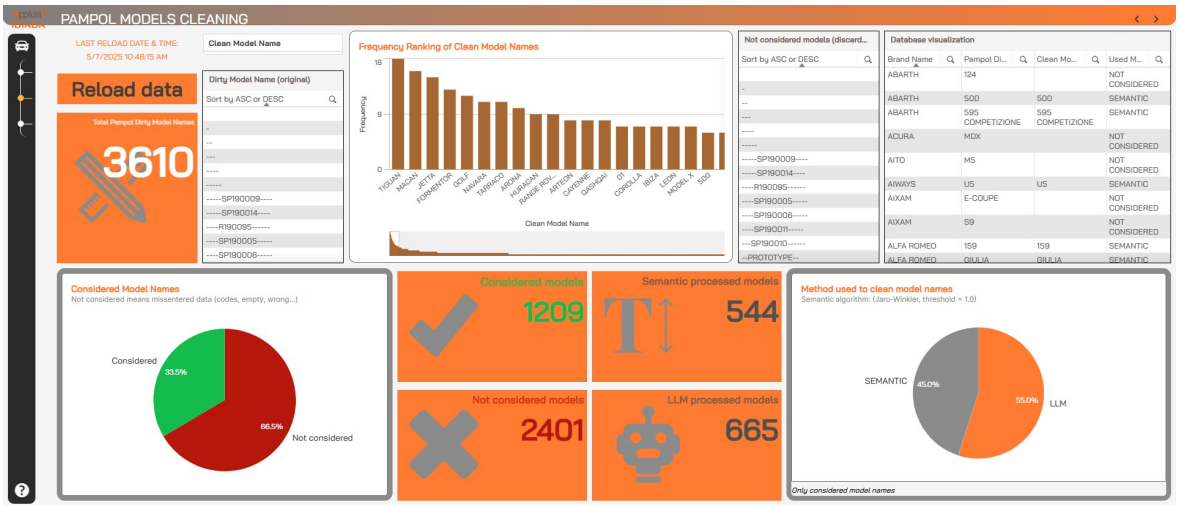


Figura 12: 1a full del *dashboard* neteja models. Full 2

Com s'aprecia a la figura 12, el full és gairebé idèntica a l'anterior, ja que pretén donar la mateixa informació bàsica. De nou, d'esquerra a dreta i de dalt a baix:

- Mostra el títol `PAMPOL MODELS CLEANING` i la data i hora de l'última actualització de les dades (5/7/2025 10:48:15 AM), en format (M/D/A).
- Nombre total de noms de models 'bruts' rebuts (3610).
- Llistat de models 'bruts' amb un botó de filtratge. Permet seleccionar un model i veure quins models 'bruts' se li han atribuït.
- Diagrama de barres que representa la distribució de freqüència dels models després del procés de neteja. L'eix vertical indica el nombre d'aparicions i l'eix horitzontal mostra els models normalitzats.
- Llistat de models no considerats.
- Visualització de la base de dades que s'està consultant.
- Gràfic circular per mostrar el percentatge de models considerats (33,5%) i no considerats (66,5%).
- Quatre indicadors visuals que quantifiquen:
  - Models considerats: 1209 registres
  - Models processats mitjançant algorisme semàntic: 544 registres
  - Models no considerats: 2401 registres
  - Models processats mitjançant LLM: 685 registres
- Gràfic circular per mostrar el percentatge de models considerats netejats amb algorisme semàntic (45,0%) i LLM (55,0%).

2. Pampol models cleaning

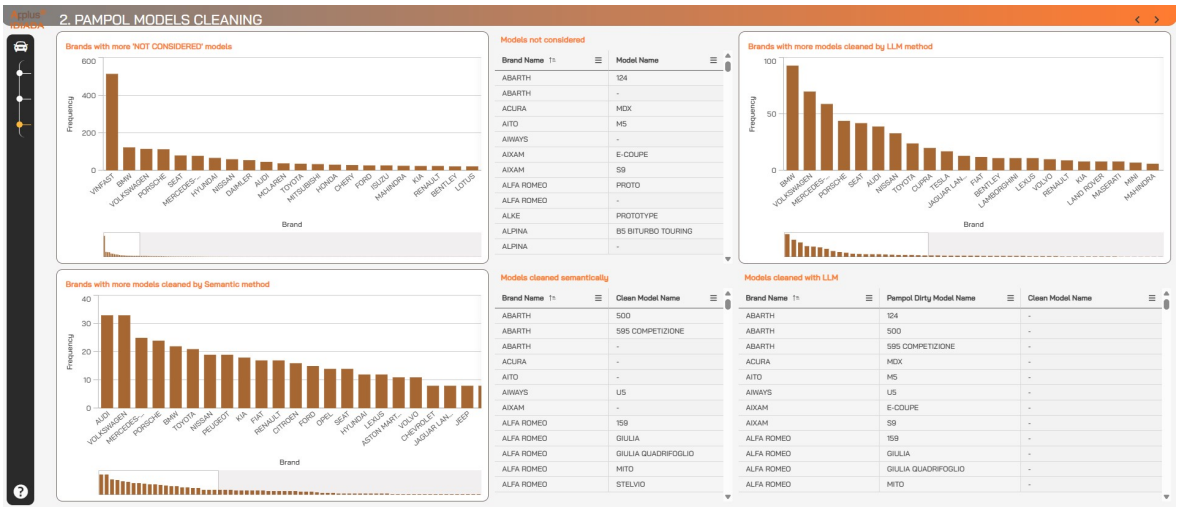


Figura 13: 2n full del *dashboard* neteja models. Full 3

Com s'aprecia a la figura 13, aquest full trenca amb l'estructura descrita fins ara, ja que l'objectiu d'aquest full és permetre una ràpida visualització d'alguns punts d'anàlisi i no una visió més general com en el cas anterior. D'esquerra a dreta i de dalt a baix:

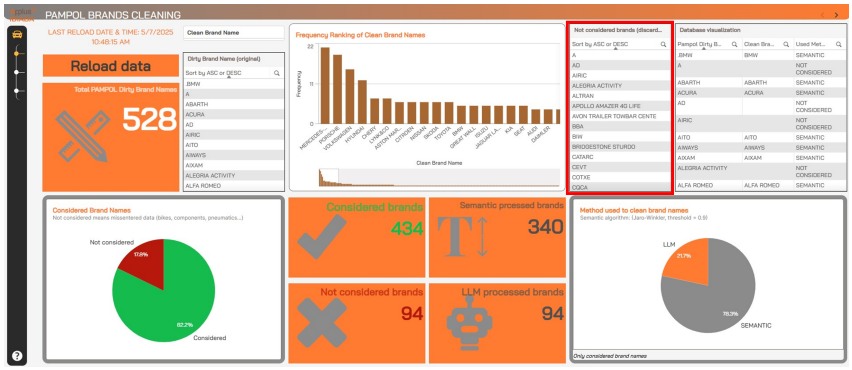
- Diagrama de barres que mostra les marques amb major nombre de models descartats pel procés de neteja. L'eix vertical mostra la freqüència i l'horitzontal les diferents marques ordenades per quantitat.
- Taula de models no considerats.
- Diagrama de barres que mostra les marques amb major nombre de models netejats amb LLM. L'eix vertical mostra la freqüència i l'horitzontal les diferents marques ordenades per quantitat.
- Diagrama de barres que mostra les marques amb major nombre de models netejats amb algorisme semàntic. L'eix vertical mostra la freqüència i l'horitzontal les diferents marques ordenades per quantitat.
- Taula de models netejats semànticament.
- Taula de models netejats amb LLM.

Valoració dels resultats

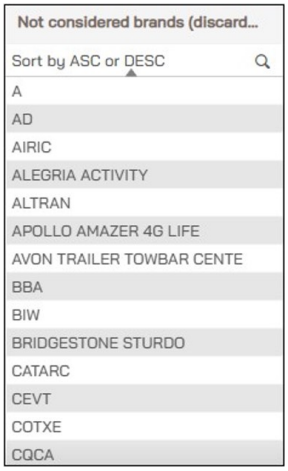
Marques

En les marques, hi ha dos conceptes importants d'estudiar per valorar la qualitat del procés: si les marques que s'estan quedant fora son correctament descartades i quines són les marques que més varietat han tingut en la seva escriptura.

El primer dels conceptes és indispensable, ja que es recolza en un dels pilars generadors d'aquest projecte. Si s'estan descartant marques que són correctes, el mestre final serà incomplet i, per tant, no funcional.



(a) Visualització taula marques descartades al *dashboard*. Full 1



(b) Taula marques descartades

Figura 14: Marques descartades

Per fer l'estudi d'aquests casos, es visualitza la taula de marques descartades mostrada a la figura 14. La figura 14b només mostra els primers elements de la taula, ordenada alfabèticament.

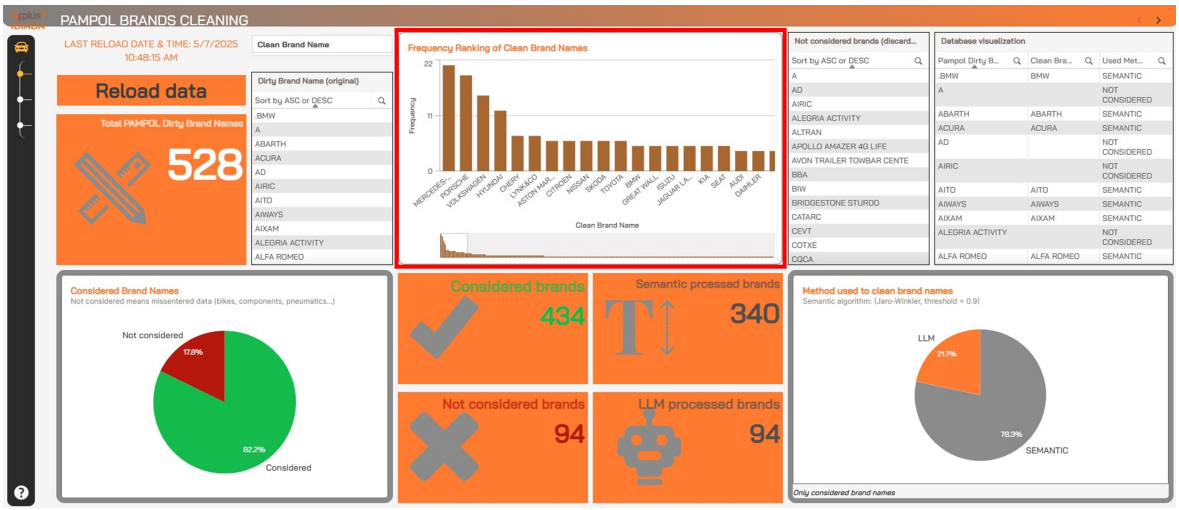
Així i tot, es poden veure quins són els motius d'exclusió d'aquestes, ja que aquests acaben extrapolant-se a la resta de casos:

- Sense sentit: A, AD, COTXE
- Components mal categoritzades a la base de dades d'origen: APOLLO AMAZER 4G LIFE i BRIDGESTONE STURDO (pneumàtics), BIW (etapa en la fabricació d'un cotxe, *Body in white*) o AVON (remolcs)
- Empreses que modifiquen vehicles amb diferents finalitats (serveis d'emergència, camperitzacions...): AIRIC, ALEGRIA ACTIVITY
- Altres centres de certificació: CATARC, CEVT

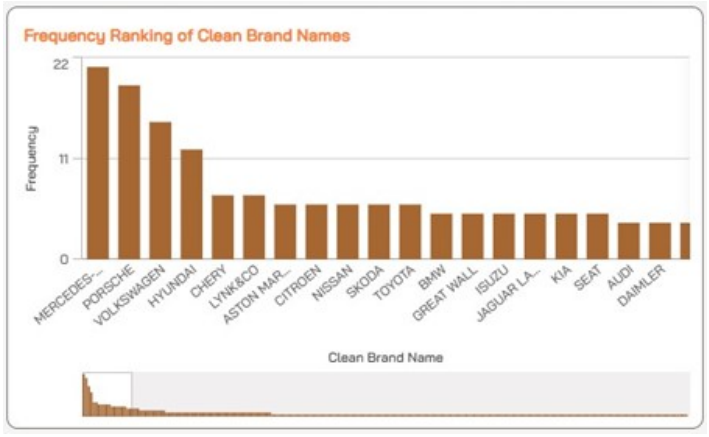
De la figura, l'únic cas que presenta dubte és BBA (*BMW Brillance Automotive*). Aquesta marca és la comercialitzadora de BMW a la Xina, però en el procés el model LLM la va descartar, ja que el model que va rebre que l'acompanyava era U12, que no correspon a cap model BMW.

A la resta de la taula hi trobem altres causes, a banda de les ja mencionades, com per exemple FCA (*Fiat Chrysler Automobiles*), PSA o STELLANTIS, descartades per ser grups de fabricants i no haver pogut extreure la informació de la marca de la llista de models o RINNEN, que és una empresa de transports, però que no és qui fabrica els camions que usa.

El segon concepte que val la pena estudiar és quines marques han resultat de netejar més varietat de marques 'brutes'. Es pot saber el motiu pel qual aquestes marques han presentat aquesta varietat i això permet conèixer millor la base de dades d'origen. Aquest estudi es pot fer a partir del diagrama de barres mostrat a la figura 15.



(a) Visualització diagrama de barres al *dashboard*. Full 1

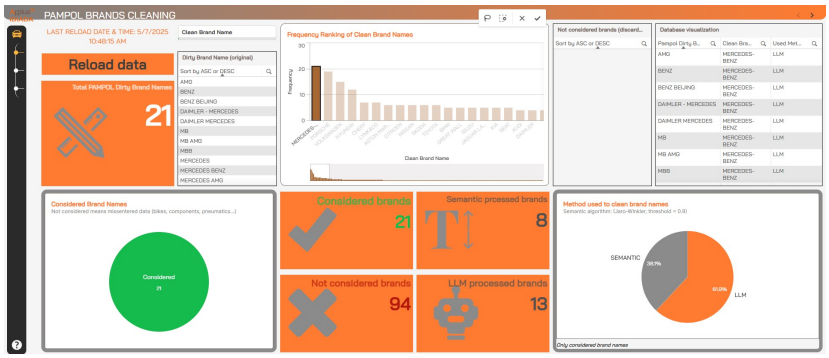


(b) Diagrama de barres

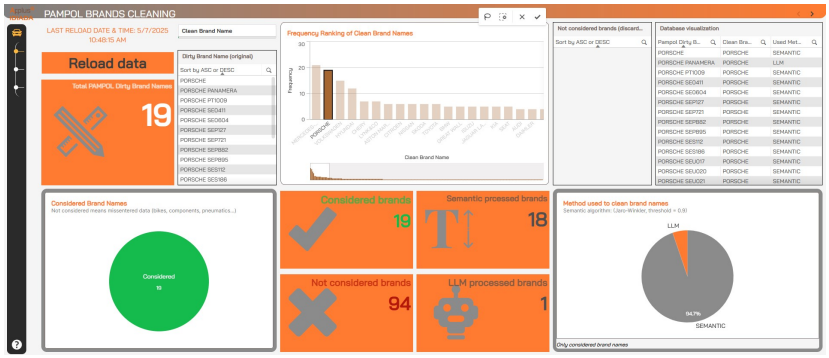
Figura 15: Marques més repetides

Com es veu les marques més repetides són MERCEDES-BENZ, PORSCHE, VOLKSWAGEN i HYUNDAI. Ara, el *dashboard* interactiu permet clicar sobre una de les barres i obtenir informació sobre aquesta, veure figura 16.

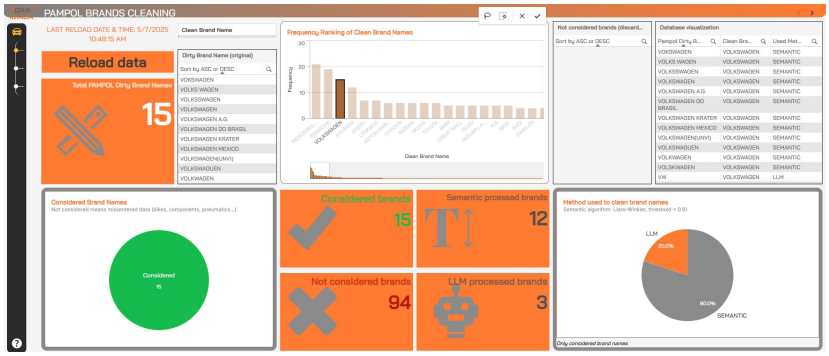
Posant ara el focus sobre la figura 16a, hom pot apreciar que hi ha hagut un total de 21 noms 'bruts' els quals han acabat estant assignats a MERCEDES-BENZ. A la llista de noms bruts s'hi poden trobar noms com: AMG, BENZ, MB, MERCEDES, MERCEDES BENZ, etc. A més, al gràfic circular de baix a la dreta podem veure que un $61,9\%$ dels noms han estat netejats amb el model LLM, mentre que el $38,1\%$ restant s'ha netejat amb l'algorisme semàntic.



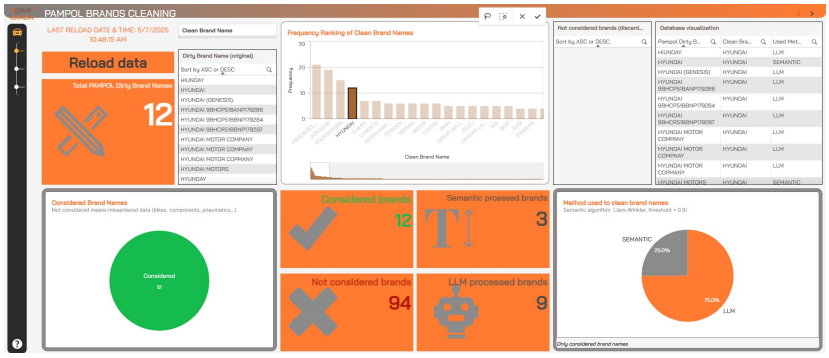
(a) *Dashboard* per marca = 'MERCEDES-BENZ'



(b) *Dashboard* per marca = 'PORSCHE'



(c) *Dashboard* per marca = 'VOLKSWAGEN'



(d) *Dashboard* per marca = 'HYUNDAI'

Figura 16: *Dashboard* filtrat per les marques amb més variabilitat. Full 1

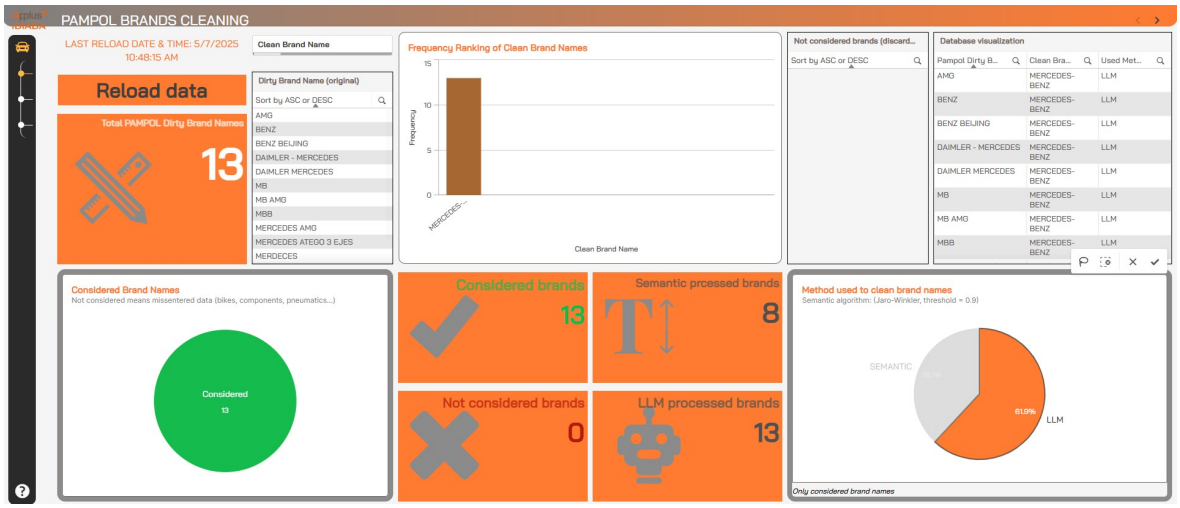


Figura 17: *Dashboard* filtrat amb marca = 'MERCEDES-BENZ' i mètode = 'LLM'. Full 1

De fet, si es clica sobre la part del gràfic de LLM, es tornarà a actualitzar el *dashboard* i veurem només la llista de noms 'bruts' netejats amb aquest mètode (el mateix passaria si ho féssim amb el SEMANTIC). Això es mostra a la figura 17.

Ara a la llista veiem, per exemple, com ha desaparegut MERCEDES BENZ, que clarament ha estat netejat amb l'algorisme semàntic pel fet que únicament hi ha l'error tipogràfic de l'absència del guió '-' intermedi.

A la figura 16b, podem apreciar que la llista de noms 'bruts' és protagonitzada per termes que inclouen PORSCHE + un codi de 6 dígits (3 lletres i 3 números). Aquest codi fa referència al nom que els tècnics utilitzen per referir-se a aquell model. Així i tot, com que comença per PORSCHE i, com ja s'ha comentat prèviament, l'algorisme semàntic de Jaro-Winkler afavoreix les coincidències en l'inici de les cadenes, per un impactant $94,7\%$ dels noms és suficient amb l'algorisme semàntic. El $5,4\%$ restant inclou únicament el nom PORSCHE PANAMERA. Aquí són 8 dígits (en comptes de 6 que tenia el codi) extres després de la paraula PORSCHE el qual és suficient per no passar el *threshold* semàntic. A més, aquest és un cas que es repeteix diversos cops, on el camp 'marca' no només inclou la marca sinó també el model, un error en l'entrada del registre a la base de dades.

La figura 16c mostra clarament els casos que és capaç de tractar el semàntic: el $80,0\%$ representa als noms que comencen per VOLKSWAGEN, mentre que el $20,0\%$ restant inclou termes com VW, VW GOLF, etc. Termes que fàcilment es poden atribuir a VOLKSWAGEN, però que clarament no passen el *threshold* de l'algorisme semàntic, ja que la similitud de les cadenes és molt baixa.

Per acabar, la figura 16d, mostra que per aquesta marca ha predominat el model LLM. Malgrat això, l'algorisme semàntic ha netejat algun error tipogràfic com HYUNDAY. Els altres termes inclouen massa dígits després de la paraula HYUNDAI que rebaixen massa la puntuació del semàntic. Un cas curiós d'aquesta marca és que el model LLM ha interpretat correctament la fabricadora a la Xina, BEJING HYUNDAI, escrita en xinès.

En conclusió, l'estudi de la neteja de marques ha permès, per una banda, veure com s'estan descartant correctament les marques que no ho són (estaven mal categoritzades o eren males entrades) i, per l'altra, comprendre quin tipus de noms 'bruts' són els que es netegen mitjançant l'algorisme semàntic (principalment errors tipogràfics, encara que com en el cas de PORSCHE també s'ha aconseguit netejar el codi que utilitzen els tècnics) i quins ho fan amb el model LLM (alguns de l'estil VW (VOLKSWAGEN) o noms escrits amb tipografia xinesa, per exemple).

Models

De nou, els models presenten un repte major a causa de la major variabilitat. Per tant, aquí l'objectiu és doble: comprovar que els models que s'estan relacionant ho fan correctament i, d'altra banda, pels que es descarten, identificar-ne els motius, cosa que permet saber si passa perquè ha de ser així (models mal entrats, codis de tècnics...) o si és per algun aspecte millorable dels mestres generats.

Primerament, es comprovaran les coincidències. Les coincidències mitjançant algorisme semàntic no són de valor d'estudi, ja que, com s'ha explicat amb anterioritat, el *threshold* utilitzat ha estat 1 i, per tant, només agafa coincidències exactes.

Altra vegada, podem filtrar el *dashboard* per veure els resultats de la neteja amb model LLM. Tal com es veu a la figura 18, 665 noms 'bruts' s'han netejat amb aquest mètode, representant el 55% del total. Amb el diagrama de barres central podem veure els noms que més cops s'han repetit i, per tant, els que han presentat més variabilitat en la base de dades original.

El primer que es veu és que tots ells són models i prou, és a dir, no contenen submodels ni subsubmodels. Això s'espera així, ja que la major part dels cops que un cotxe entra a l'empresa per ser testejat no és important quina variant del model és, perquè el resultat serà vàlid per a tots ells, i, per tant, a l'hora d'omplir el camp model s'emplena de la forma més general possible.

Figura 18: *Dashboard* filtrat amb mètode = 'LLM'. Full 2

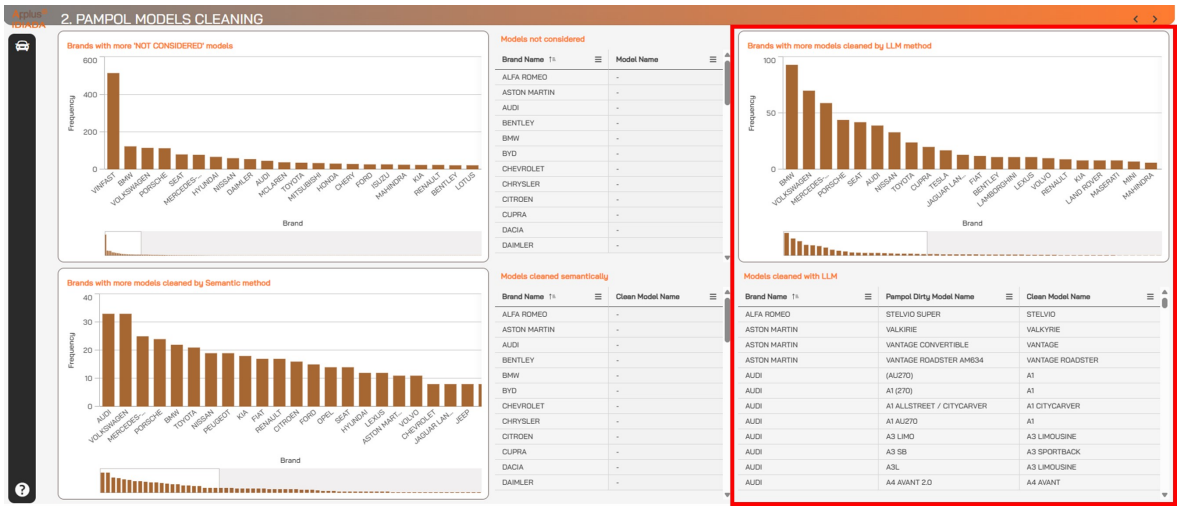


Figura 19: Secció estudi procés LLM models. Full 3

A la figura 19, es remarca en quina secció es fa l'anàlisi d'aquest cas. Primerament, a la part superior, es veu quines són les marques de les quals més models han estat relacionats amb aquest mètode. Destaquen BMW, VOLKSWAGEN, MERCEDES-BENZ i PORSCHE. 3 de les 4 també han sortit com les més repetides en l'anàlisi de marques, el qual indica que són marques amb força activitat dins l'empresa.

A la taula de baix hi podem trobar els models netejats. Aquesta taula serà de gran utilitat per veure l'eficàcia del mestre. Hi ha dos casos, el ALFA ROMEO STELVIO SUPER i el ASTON MARTIN VANTAGE CONVERTIBLE que els hi passa el mateix: a l'hora de la neteja se'ls hi suprimeix el submodel (SUPER i CONVERTIBLE, respectivament). El motiu, però, és diferent.

Pel cas de l'ALFA ROMEO, al mestre sí que hi trobem el model SUPER, veure figura 20. El problema és que, com s'aprecia, el codi no ha pogut separar el submodel SUPER de la resta de la cadena (TB/TD AWD AUTO) que representaria el subsubmodel. Això és així perquè el codi necessita rebre almenys un cop, com entrada, ALFA ROMEO SUPER, sense res més, per fer la partició. Si això no fos així, el codi presentaria problemes amb models de nom compost (com pot ser RANGE ROVER), ja que consideraria RANGE model i ROVER submodel.

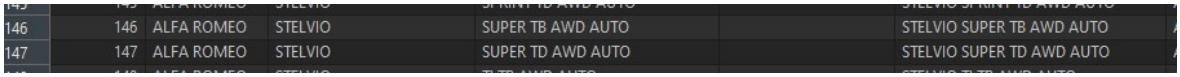


Figura 20: ALFA ROMEO SUPER al mestre

En canvi, pel cas de l'ASTON MARTIN, el model VANTAGE sí que apareix al mestre, però no hi ha cap registre amb CONVERTIBLE com a submodel, tal com es pot comprovar a la figura 21. Això és així pel fet que no ha entrat cap registre sota aquest nom des de la base de dades de referència (la de la EEA).

Un altre cas molt curiós que es pot apreciar a la taula de la figura 19 és com tracta l'agent LLM el model A1. Com es pot veure, hi ha una entrada amb nom 'brut' A1 AU270, una sota el nom A1 (270) i una altra sota el nom (AU270). Amb tota aquesta informació, és capaç de decidir que tots ells són en realitat el mateix model, l'A1. Si l'agent no rebés els models en *batches*, sinó individualment, aquest no hagués estat capaç de reconèixer (AU270) com un A1.

227	ASTON MARTIN	VANTAGE			VANTAGE	M
228	ASTON MARTIN	VANTAGE	AMR		VANTAGE AMR	A
230	ASTON MARTIN	VANTAGE	F1	EDITION	VANTAGE F1 EDITION	A
231	ASTON MARTIN	VANTAGE	ROADSTER		VANTAGE ROADSTER	M
232	ASTON MARTIN	VANTAGE	ROADSTER	F1 EDITION	VANTAGE ROADSTER F1 EDITION	A
233	ASTON MARTIN	VIRAGE			VIRAGE	M

Figura 21: ASTON MARTIN VANTAGE al mestre

Finalment, també podem apreciar com converteix SB en SPORTBACK, LIMO a LIMOUSINE, etc. Aquest tipus de conversions són els casos on l'agent LLM funciona millor. De fet, és el mateix tipus de neteja discutit anteriorment en la secció d'anàlisi de marques passant de VW a VOLKSWAGEN.

El model LLM també ha aconseguit netejar casos on el camp model inclou no només el nom comercial sinó també un codi alfanumèric, deixant només el nom comercial, com es pot veure a la figura 22.

Models cleaned with LLM

Brand Name	Pampol Dirty Model Name	Clean Model Name
NISSAN	MICRA EKE696	MICRA
NISSAN	NAVARA	-
NISSAN	NAVARA (FRONTIER)	NAVARA
NISSAN	NAVARA FRONTIER	NAVARA
NISSAN	NAVARA H60A	NAVARA
NISSAN	NAVARA TZA K66	NAVARA
NISSAN	NAVARA TZC743E6	NAVARA
NISSAN	NAVARA TZC745E6	NAVARA
NISSAN	NAVARA TZC757E6	NAVARA
NISSAN	NAVARA TZC772EG	NAVARA
NISSAN	NAVARRA	NAVARA
NISSAN	NISSAN P33B CVT VC-LOT	-

Figura 22: Exemple neteja codis alfanumèrics

Tot i això, hi ha casos on el model LLM també s'equivoca. Per exemple, a la figura 23 podem veure'n un cas. Al mestre hi ha el HYUNDAI SANTA FE, però no el HYUNDAI SANTA CRUZ, ja que aquest no es comercialitza a Europa. Aquesta és una limitació provinent del fet que la base de dades de referència sigui únicament la EEA (europea). El model LLM, però, erra atribuint-li SANTA CRUZ com a nom net, perquè aquest és un nom corresponent a un model diferent, però, a causa de la similitud en els noms, s'equivoca.

En conclusió, el model LLM és veritablement capaç de netejar casos on s'ha escrit el nom del model usant alguna abreviatura i aquest tipus de casuístiques. Amb tot i això, no és 100% fiable, sinó que en algun erra relacionant. Malgrat que l'important per aquest estudi no és com de bé treballa l'agent LLM, aquest té una influència capital en el projecte i és essencial entendre'n les limitacions per garantir un bon resultat final.

Models cleaned with LLM

Brand Name	Pampol Dirty Model Name	Clean Model Name
HONDA	CR-V AWD	CR-V
HONDA	CR-V HIBRID	CR-V HYBRID
HONDA	CRV	CR-V
HONDA	HRV	HR-V
HONDA	JAZZ CROSSTAR	JAZZ
HYUNDAI	BI3 - I20	I20
HYUNDAI	I 10	I10
HYUNDAI	SANTA CRUZ	SANTA FE
HYUNDAI	TUSCON	TUCSON
ISUZU	D MAX	D-MAX
ISUZU	DMAX	D-MAX
IVECO	DAILY 70C	DAILY

Figura 23: Neteja del HYUNDAI SANTA CRUZ

Per altra banda, en quant al mestre, s'ha vist que aquest presenta una limitació a l'hora de dividir el camp en model, submodel i subsubmodel consistent en el fet que no és capaç de separar-los si no rep almenys un registre de cada.

Així i tot, tenint en compte que la base de dades s'omple de forma manual, sense cap restricció es pot considerar que la neteja ha estat més que correcte, ja que s'han estandarditzat molts noms amb variacions tipogràfiques o inclús casos més avançats com el de l'AUDI (AU270).

Pel que fa a la part de models no considerats, a la figura 24 es marca la secció del *dashboard* que els hi correspon.

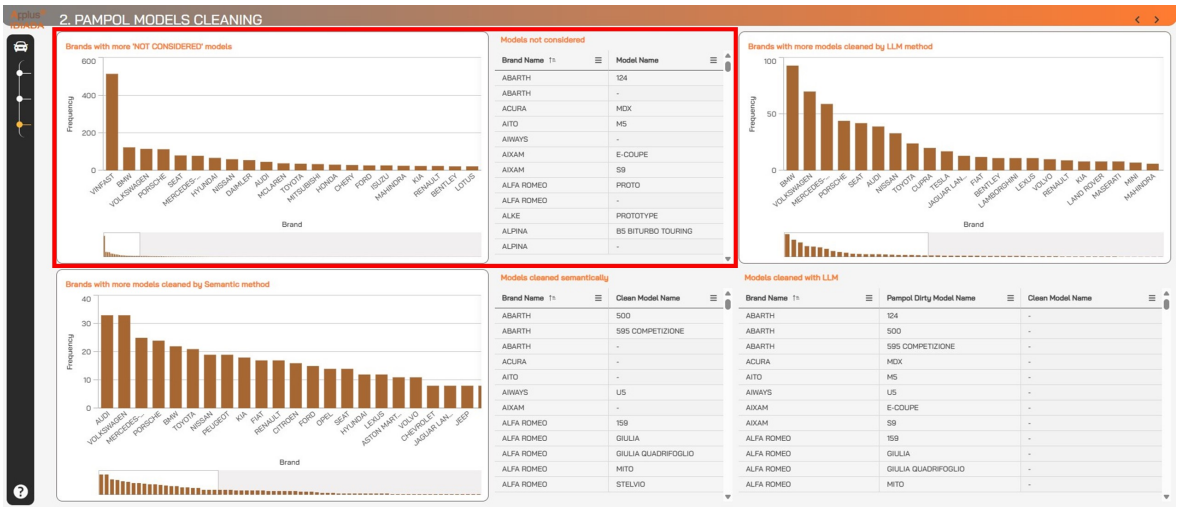


Figura 24: Secció estudi procés NOT CONSIDERED models. Full 3

En aquest cas, destaca clarament una marca sobre la resta, VINFAST. VINFAST és una marca vietnamita que va començar a comercialitzar els seus vehicles a Europa al 2024 [20]. Les dades de la EEA amb les quals s'ha generat el mestre arriben al 2022, per la qual cosa, de nou, és un cas en què el problema és que la base de referència sigui

únicament europea. A més, VINFAST és un client molt potent dins l'empresa [21], per la qual cosa és normal que aparegui molts cops a la base de dades de Pampol que s'està netejant.

Un altre tipus de cas on la limitació és deguda al fet que la base de dades de referència sigui europea és quan una marca produeix cotxes en tots els mercats, però es reserva alguns models per mercats específics, tal com s'ha vist amb el HYUNDAI SANTA CRUZ. En aquell cas, l'agent LLM ha considerat malament que es tractava d'un HYUNDAI SANTA FE i l'ha netejat, però aquesta taula inclou casos, com VOLKSWAGEN ATLAS (només comercialitzat a Amèrica i Xina), que descarta.

Un altre cas de marca la qual no se n'estan considerant els models perquè aquests no es comercialitzen a Europa, que es pot apreciar al diagrama de barres, és CHERY, una marca xinesa.

A la taula es pot apreciar un altre patró recurrent entre els models no considerats: els prototips. Concretament, a la captura s'hi veuen ALFA ROMEO PROTO i ALKE PROTOTYPE. Evidentment, a IDIADA s'hi testegen i homologuen cotxes encara no comercialitzats i, per tant, tampoc recollits a la base de dades de la EEA. Així doncs, els models on s'indica que és un prototip i també els models que són codis alfanumèrics (com que no tenen encara un nom comercial, en fases prèvies s'utilitzen aquestes nomenclatures), no són considerats durant la neteja.

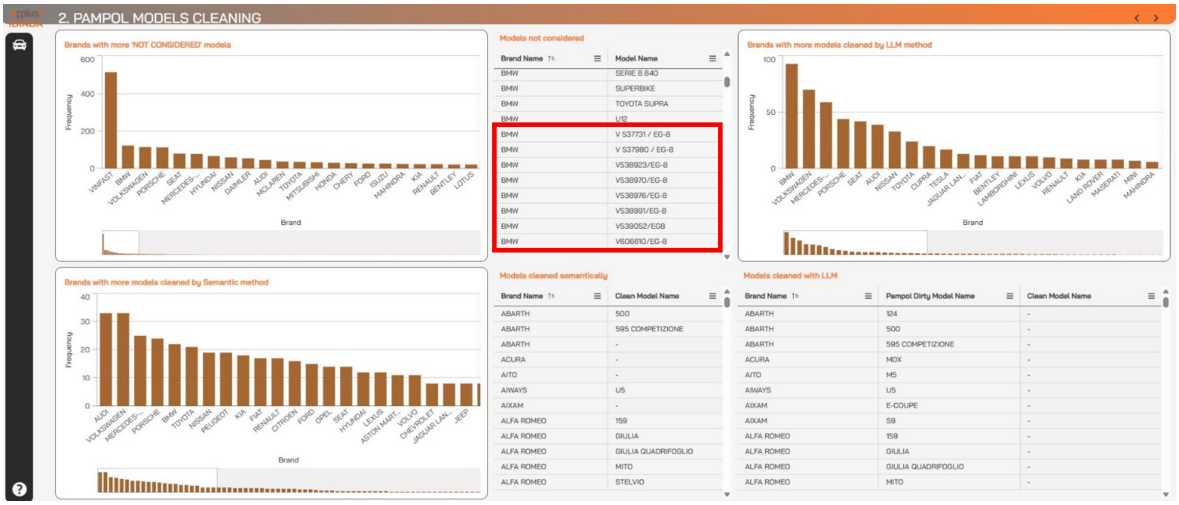


Figura 25: Exemple models omplerts amb codis alfanumèrics

A la figura 25 apreciem un cas en què, en comptes d'aparèixer el nom comercial al camp model, hi apareix un codi alfanumèric. Pot ser per dues possibles causes, que es tracti també d'un prototip o que sigui un model ja comercialitzat, però que qui va registrar l'entrada va fer servir un codi intern. Això també és un fet recurrent a la taula. En aquest cas, com ja s'ha comentat, aquests casos es descarten, però si la finalitat de la neteja fos una altra, es podria tenir un `output` diferent.

Un altre patró recurrent és el cas d'entrades 'sense sentit', és a dir, entrades com les que es marquen a la figura 26, CUPRA CUPRA o CUPRA FAMILIAR. Evidentment, aquests no són models de la marca CUPRA i, per tant, s'estan descartant correctament.

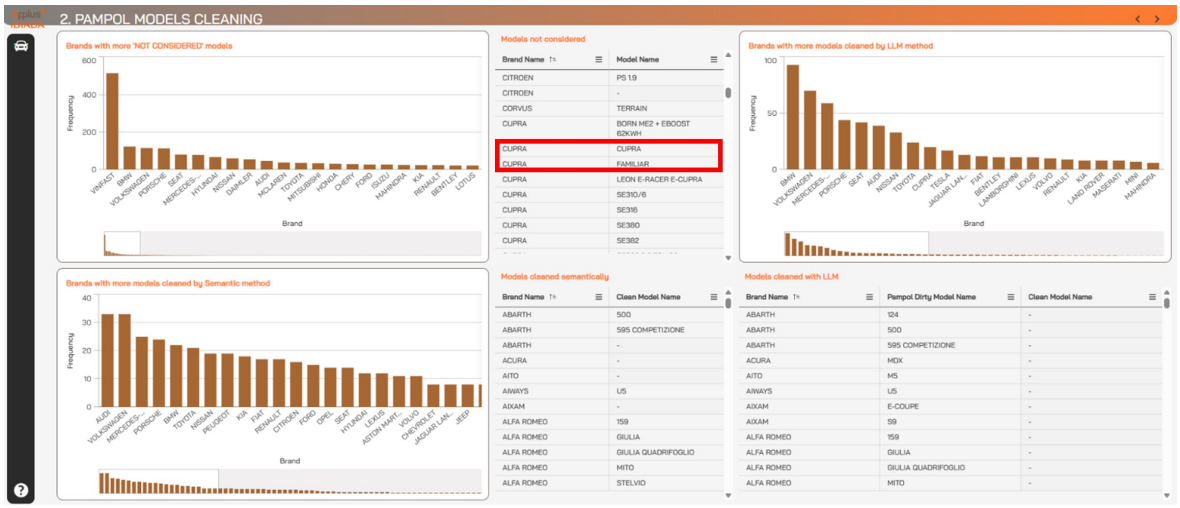


Figura 26: Exemple models sense sentit

L'últim dels patrons identificats són motos mal categoritzades de marques que sí que són al mestre, com BMW, HONDA, etc., que també acaben incrementant el nombre de models no considerats. Veure figura 27. De nou, segons la finalitat de la neteja, es podria optar per recategoritzar els registres, per exemple.

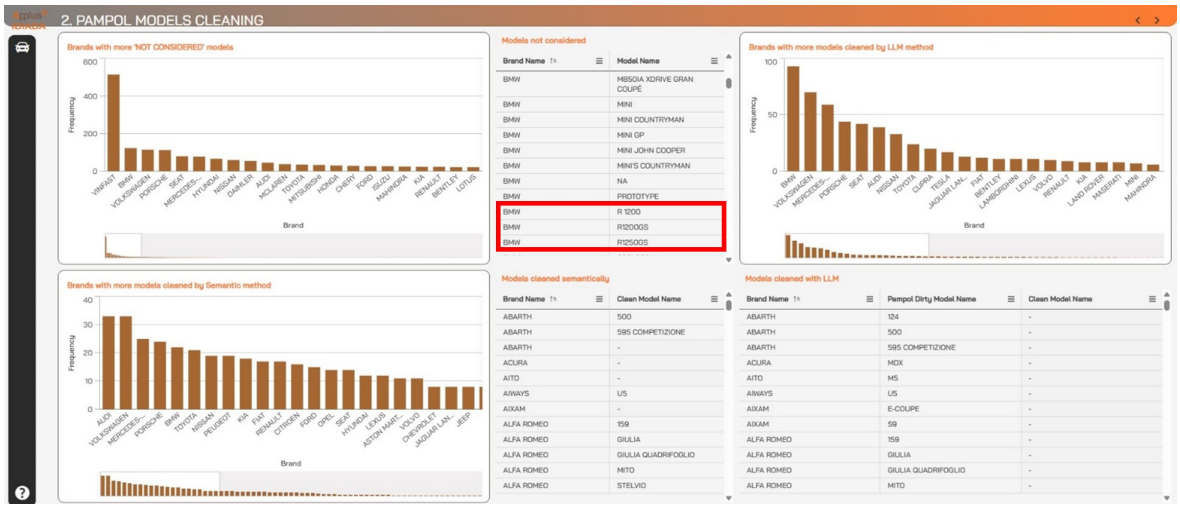


Figura 27: Exemple d'entrades mal categoritzades

Finalment, també hi ha casos de models que no s'estan considerant perquè no hi són a mestre. És important recordar que a l'agent LLM se li van donar instruccions clares de no perdre informació, per la qual cosa, quan un model a la base de dades de Pampol conté informació no present al mestre, per exemple com passa a la figura 28, on NISSAN QASHQAI sí que hi és al mestre però el submodel E-POWER no. En aquest cas concret, és perquè aquest va sortir l'any 2022 i, per tant, no ha estat usat per la generació del mestre, però hi ha casos on el motiu és falta de completesa del mestre.

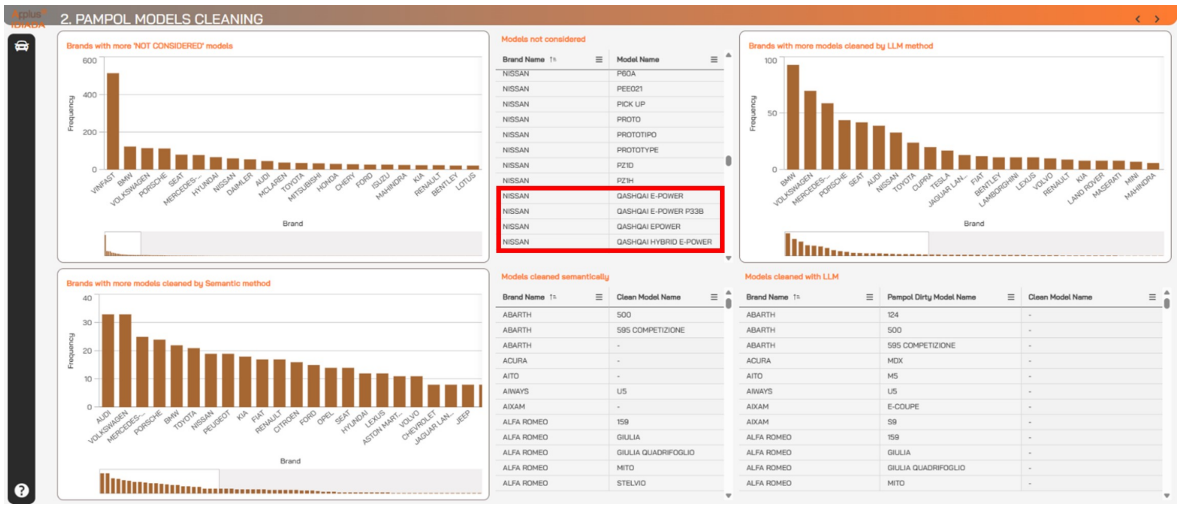


Figura 28: Exemple de models no registrats al mestre

Per tant, pels casos no considerats tenim diverses consideracions: la primera és que s'evidencia la limitació de que la base de dades de referència (EEA) sigui únicament europea, produint marques les quals cap model és considerat i marques que comercialitzen models només per altres mercats i aquests tampoc són considerats.

La segona és el fet que la base de dades de Pampol estigui plena de codis alfanumèrics corresponents a prototips o a nomenclatures internes, incrementant significativament el nombre de models no considerats. La tercera són vehicles mal categoritzats i que, per tant, no estan inclosos en el mestre i es descarten.

Finalment, es considera que hi ha casos on el mestre es mostra incomplet, sobretot en les capes de submodels i subsubmodels. Així i tot, la gran major part de models descartats ho han estat de forma correcta.

La taula 6 mostra un resum dels patrons trobats durant l'anàlisi de models, explicitant els patrons trobats i el motiu de l'anomalia:

Patró	**Motiu**	**Origen**	**Relació mestre**	**Exemple**	**Resultat**
Falta de jerarquització	No ha entrat cap STELVIO SUPER i per tant no s'ha separat al mestre	Funció del *script* que jerarquitza els models necessita que entri com a mínim un cop per separat	Sí	STELVIO SUPER tenint al mestre STELVIO SUPER TB/TD AWD AUTO	Models no considerats o mal relacionats
Models no comercialitzats a Europa són descartats	No hi són al mestre	Base de dades de referència (EEA) és únicament Europa	Sí	VOLKSWAGEN ATLAS, VINFAST	Models no considerats
Codis alfanumèrics	Prototips	A IDIADA hi entren cotxes no comercialitzats encara	No	ALKE PROTOTYPE	Models no considerats
	Codis interns	Base de dades interna	No	BMW V538976/EG-8	Models no considerats
Sense sentit	Entrada manual	La base de dades s'emplena manualment	No	CUPRA FAMILIAR	Models no considerats
Males categoritzacions	Vehicles i components categoritzats malament	Categorització manual	No	BMW R1200GS	Models no considerats

Taula 6: Taula de patrons trobats durant l'anàlisi de models

Conclusions finals de l'anàlisi

L'objectiu d'aquesta anàlisi era estudiar l'eficàcia del mestre, per tal de trobar tant els punts on aquest és més fort com aquells on hi ha marge de millora. La neteja s'ha fet sobre una base de dades real i, per tant, presentava les imperfeccions esperades derivades d'un sistema en producció sotmès a l'ús diari. El fet que els registres s'entrin manualment en un camp de text lliure, sense restriccions ni validacions automàtiques, també fa que molts cops els camps continguin informació errònia. En conseqüència, el procés no només ha de normalitzar els noms de models sinó també ser capaç d'identificar i descartar les entrades dolentes.

Amb tot i això, es pot valorar positivament el procés. A continuació, es presenten les principals conclusions:

Eficàcia general del procés

El procés de neteja ha demostrat ser notablement eficaç, especialment considerant que treballava sobre una base de dades real amb les imperfeccions pròpies de l'ús quotidià. El mestre ha aconseguit estandarditzar correctament un alt percentatge de marques (82,2%), amb especial èxit en aquelles més freqüents com MERCEDES-BENZ, PORSCHE i VOLKSWAGEN.

Complementarietat dels mètodes de neteja

S'ha confirmat la complementarietat entre l'algorisme semàntic i el model LLM:

- L'algorisme semàntic (78,3% de les marques considerades) ha excel·lit en la correcció d'errors tipogràfics i casos amb petites variacions.
- El model LLM (21,7% de les marques i 55% dels models) ha demostrat ser especialment útil per casos més complexos com abreviatures (VW→VOLKSWAGEN, SB→SPORTBACK), marques amb noms molt diferents (AUTOMOBILI LAMBORGHINI→LAMBORGHINI) o codis alfanumèrics.

Limitacions identificades

S'han identificat diverses limitacions que afecten l'eficàcia del mestre:

1. Base de dades de referència: En basar-se exclusivament en dades europees (EEA), el mestre no reconeix models comercialitzats només en altres mercats (com VINFAST o VOLKSWAGEN ATLAS). Marques sí, ja que el procés de generació del mestre de marques no es basava únicament en la EEA.
2. Jerarquització de models: El sistema requereix que cada nivell de la jerarquia (model, submodel, subsubmodel) aparegui almenys un cop de forma independent per poder separar-lo correctament, com en el cas d'ALFA ROMEO STELVIO SUPER.
3. Prototips i models nous: El mestre no pot reconèixer prototips o models molt recents que encara no figurin a la base de dades de referència, com el NISSAN QASHQAI E-POWER (llançat el 2022).

Gestió d'entrades problemàtiques

El sistema ha gestionat adequadament les entrades problemàtiques:

- Ha descartat correctament entrades sense sentit (CUPRA FAMILIAR), codis alfanumèrics per a prototips, vehicles mal categoritzats i noms de grups fabricants.
- Ha netejat satisfactòriament casos on el camp model contenia codis alfanumèrics juntament amb el nom comercial.

Valoració global i perspectives

En conjunt, malgrat les limitacions identificades, el mestre ha demostrat ser una eina robusta i eficient per a la normalització de marques i models de vehicles. La major part dels registres no considerats ho han estat correctament, confirmant la capacitat del sistema per identificar i filtrar dades inconsistents o errònies.

Les àrees de millora més destacables són:

- Ampliació de la base de dades de referència per incloure models d'altres mercats
- Refinament del procés de jerarquització per facilitar la separació en model, submodel i subsubmodel

Finalment, cal destacar que l'efectivitat del procés de neteja depèn en gran manera de l'objectiu final. En aquest cas, l'enfocament ha prioritzat la precisió i la qualitat de les dades, però segons la finalitat podrien considerar-se estratègies diferents per gestionar casos específics com els vehicles mal categoritzats o els prototips.

6 Conclusions

En aquesta fase final del Treball de Fi de Grau, puc afirmar que l'experiència ha estat altament enriquidora tant en l'àmbit tècnic com personal. El projecte ha suposat un repte complet i transversal, en què he tingut l'oportunitat de participar en totes les etapes del cicle de vida de la dada, des de la captura i processament mitjançant processos ETL fins a la seva explotació analítica.

Si bé és cert que en l'inici del projecte em feia patir no estar prou preparat tècnicament (ja que, com s'ha vist, el projecte requereix un alt grau de coneixement de programació i aquest no és un pilar principal dels meus estudis) durant el desenvolupament del projecte he anat adquirint confiança i, a escala d'aprenentatge, ha acabat esdevenint una de les parts més valuoses.

Per exemple, el desenvolupament de la *pipeline* d'automatització sobre l'entorn AWS m'ha permès familiaritzar-me amb serveis *cloud* avançats com Lambda, Step Functions i S3, integrant-los mitjançant scripts Python per construir una arquitectura escalable i mantinguda. Aquest enfocament m'ha ofert una visió pràctica de com es desenvolupen fluxos de dades en entorns empresarials reals.

Des del punt de vista analític, la fase final del projecte ha inclòs el disseny i creació de *dashboards* a Qlik per visualitzar els resultats de neteja i normalització dels mestres. Aquesta part ha estat clau per validar de forma intuïtiva i visual la qualitat del procés, i m'ha permès un primer contacte amb aquest tipus d'eines de BI.

Pel que fa a formació, aquest projecte ha estat especialment valuós perquè, malgrat que el desenvolupament ha estat centrat principalment en tasques de programació, he pogut aplicar-hi competències transversals adquirides durant el Grau en Enginyeria Matemàtica i Física. La capacitat d'analitzar problemes complexos, estructurar solucions de manera lògica i afrontar reptes amb un pensament crític rigorós ha estat clau per entendre, dissenyar i desplegar tot el sistema.

Tot i que el grau no inclou un enfocament específic en programació avançada o enginyeria de dades, la solidesa matemàtica i la familiaritat amb entorns abstractes i metodologies de resolució de problemes han facilitat l'aprenentatge autònom de tecnologies com Python, AWS i Qlik. En aquest sentit, considero que la meva formació ha estat una base sòlida i adaptable per abordar amb èxit un projecte com aquest, alineat amb el món professional i tecnològic actual.

A més, el projecte m'ha servit per millorar habilitats en gestió de projectes i col·laboració en entorns multidisciplinaris, competències clau per a la meva futura trajectòria professional.

En definitiva, aquest Treball de Fi de Grau ha estat una oportunitat única per posar en pràctica tot l'après durant els darrers anys, i considero que ha contribuït de manera significativa a la meva formació com a enginyer matemàtic i físic amb una orientació clara cap a l'àmbit de l'enginyeria de dades.

Pròximes passes

Tot i que el sistema desenvolupat compleix amb els objectius inicials, durant el projecte s'han identificat diverses línies de millora que podrien enriquir encara més els mestres.

- Un dels avenços naturals seria la incorporació de noves bases de dades de referència, provinents de fonts oficials complementàries. Aquesta ampliació permetria millorar la cobertura del sistema. Així, es deixaria d'excloure del procés vehicles no comercialitzats a Europa i s'incrementaria la informació continguda en els mestres.
- Una altra línia de treball especialment rellevant és la millora del sistema de jerarquització dels models, submodels i subsubmodels. Com s'ha identificat durant el treball, el procés actual necessita almenys un registre per separat per tal de partir el nom comercial i jerarquitzar-lo i, malgrat que no fer-ho així implicaria problemes amb noms de models compostos, convindria investigar diferents mètodes.
- Finalment, un altre aspecte a explorar és la reducció o eliminació de la necessitat de supervisió manual sobre les sortides generades pel model de llenguatge (LLM). Actualment, tot i que l'automatització permet accelerar el procés, és necessària una validació manual en casos amb baixa certesa.

7 Consideracions ètiques i de responsabilitat social

Aquest projecte, com a part d'un entorn empresarial real i relacionat amb el tractament de dades, requereix una reflexió sobre aspectes ètics, ambientals i de responsabilitat social. A continuació, s'analitza la possible implicació del projecte en relació amb quatre indicadors específics: igualtat, medi ambient, responsabilitat social i ètica.

7.1 Igualtat

Tot i que el projecte no treballa amb dades personals identificables ni amb aplicacions orientades al gran públic, sí que inclou la creació de *dashboards* que poden considerar-se com a interfícies d'usuari final dins l'àmbit corporatiu. Per aquest motiu, és important tenir en compte la possibilitat que els algoritmes o models utilitzats puguin incorporar o amplificar algun tipus de biaix. No obstant això, en aquest cas concret, s'ha treballat exclusivament amb dades tècniques de vehicles (marques, models i especificacions), per la qual cosa no s'han detectat riscos directes de discriminació per raó de gènere.

Tanmateix, s'ha mantingut una actitud crítica i reflexiva davant l'ús de models de llenguatge de grans dimensions (LLM), ja que aquests poden contenir biaixos implícits segons les dades amb què han estat entrenats. A causa de la seva aplicació a tasques purament tècniques com la normalització de noms, s'han evitat usos interpretatius o que poguessin tenir impacte social.

7.2 Medi ambient

La perspectiva mediambiental ha estat present de forma indirecta. Com bé s'ha anat explicitant al llarg del treball, s'ha procurat minimitzar al màxim el nombre de crides al model de llenguatge de grans dimensions (LLM). Aquesta decisió no només respon a criteris d'eficiència i cost, sinó també a una consciència mediambiental: els LLM tenen un alt consum computacional, i el seu entrenament i ús comporta una despesa energètica significativa, així com un consum indirecte d'aigua per a la refrigeració dels servidors. Per aquest motiu, s'ha optat per aplicar primer tècniques semàntiques més lleugeres i fer ús del LLM només quan ha estat estrictament necessari, contribuint així a una reducció de l'impacte ecològic associat.

A més, el fet de desenvolupar processos automatitzats i optimitzats contribueix a una millor eficiència dels sistemes digitals, reduint el temps de processament i l'ús de recursos computacionals i, per tant, l'impacte energètic associat. L'ús d'infraestructura al núvol també ha facilitat una millor gestió dels recursos, ja que permet escalar el consum a la demanda real.

7.3 Responsabilitat social

El projecte contribueix directament a millorar la qualitat de les dades i la coherència dels sistemes d'una empresa multinacional com Applus+ IDIADA. La creació de taules mestres i la seva integració amb aplicacions corporatives millora l'eficiència operativa, redueix errors i facilita la presa de decisions basada en dades fiables.

Aquesta contribució indirecta a una millor qualitat dels processos d'homologació o testatge de vehicles pot considerar-se un exemple d'impacte social positiu.

7.4 Ètica

L'ús de dades i tecnologies com AWS o models de llenguatge (LLM) comporta responsabilitats ètiques. El projecte ha permès desenvolupar una actitud crítica davant l'automatització de processos, reflexionant sobre els límits d'allò que pot decidir un algorisme i allò que requereix intervenció o supervisió humana. En aquest sentit, s'ha prioritzat sempre el principi de prudència en l'ús de tecnologies potencialment opaques com els LLM.

A més, s'han tingut en compte els principis deontològics del camp de l'enginyeria i la gestió de dades: transparència, traçabilitat, fiabilitat i responsabilitat en el tractament de la informació i sempre s'ha actuat amb respecte a les normes pròpies de la comunitat universitària i professional, garantint la integritat del treball i promovent bones pràctiques en el desenvolupament de solucions tecnològiques.

Referències

[1] Applus IDIADA. Sobre applus+ idiada, Febrer 2025. https://www.applusidiada.com/global/es/about-us/inbrief

[2] Diari de Tarragona. El govern adjudica a applus el contrato de idiada por 428 millones, el doble de la licitación inicial, Febrer 2025. https://www.diaridetarragona.com/economia/el-govern-adjudica-a-applus-el-contrato-de-idiada-por-428-millones-el-doble-de-la-licitacion-inicial-JK20199066

[3] Applus IDIADA. Digital solutions, Febrer 2025. https://www.applusidiada.com/global/es/what-we-do/services/soluciones-digitales

[4] IBM. ¿qué es una api (interfaz de programación de aplicaciones)?, Abril 2025. https://www.ibm.com/es-es/topics/api

[5] Wikipedia, The free encyclopedia. Web-scraping, Abril 2025. https://en.wikipedia.org/wiki/Web_scraping

[6] Wikipedia, The free encyclopedia. Python, Maig 2025. https://en.wikipedia.org/wiki/Python_(programming_language)

[7] Pandas. Pandas documentation, Maig 2025. https://pandas.pydata.org/docs/

[8] NumPy. Numpy documentation, Maig 2025. https://numpy.org/doc/

[9] Amazon Web Services. Boto3 documentation, Maig 2025. https://boto3.amazonaws.com/v1/documentation/api/latest/index.html

[10] Wikipedia, The free encyclopedia. Sql, Maig 2025. https://en.wikipedia.org/wiki/SQL

[11] Wikipedia, The free encyclopedia. Jaro-winkler distance, Febrer 2025. https://en.wikipedia.org/wiki/Jaro-Winkler_distance

[12] Amazon Web Services. Aws documentation, Maig 2025. https://docs.aws.amazon.com/

[13] DBeaver. Dbeaver documentation, Maig 2025. https://dbeaver.io/docs/

[14] Qlik. Qlik documentation, Maig 2025. https://help.qlik.com/

[15] Wikipedia, The free encyclopedia. Levenshtein distance, Febrer 2025. https://en.wikipedia.org/wiki/Levenshtein_distance

[16] Wikipedia, The free encyclopedia. Tf-idf, Febrer 2025. https://en.wikipedia.org/wiki/Tf-idf

[17] Wikipedia, The free encyclopedia. Large language model, Maig 2025.
https://en.wikipedia.org/wiki/Large_language_model

[18] El país, El motor. De la a a la z: todas las marcas de coches del mundo, Març 2025.
https://motor.elpais.com/actualidad/de-la-a-a-la-z-todas-las-marcas-de-coches-del-mundo/

[19] Carlogos. All car brands, Març 2025.
https://www.carlogos.org/car-brands/

[20] Hibridos y Eléctricos. Vinfast llegará masivamente a europa en 2024 con hasta cinco modelos eléctricos, Abril 2025.
https://www.hibridosyelectricos.com/coches/llega-europa-suv-electrico-mas-pequeno-mas-asequible-esta-marca-no-es-china_72873_102.html

[21] VINGroup. Vinfast collaborates with applus+ idiada in safety performance testing for ev, Abril 2025.
https://www.vingroup.net/en/news/detail/2472/vinfast-collaborates-with-applus-idiada-in-safety-performance-testing-for-ev

[22] PostgreSQL. Postgresql: The world's most advanced open source relational database, Març 2025.
https://www.postgresql.org