Enric Peñalver Hill

Estudi d'implementació d'un model de classificació per a assaigs de cinemàtica i conformitat a la indústria automobilística

Treball de Fi de Grau
Dirigit pel Dr Roger Guimerà Manrique

Grau d'Enginyeria Matemàtica i Física

UNIVERSITAT ROVIRA i VIRGILI

Tarragona
2025

Agraïments

Vull expressar el meu sincer agraïment al professor i tutor del projecte, el Dr. Roger Guimerà Manrique, per la orientació proporcionada així com pels coneixements en tècniques avançades d'aprenentatge automàtic, els quals han estat clau per afrontar les dificultats d'aquest estudi.

També voldria agrair especialment a l'equip d'enginyers del subdepartament de K&C, Jorge Muñoz i Diego Riestra, per la seva col·laboració, suport tècnic i per compartir el seu coneixement pràctic sobre el funcionament dels assaigs experimentals i interpretació de les dades, contribuint de manera decisiva al desenvolupament de l'estudi i de l'eina final.

Sense la implicació i suport de les persones mencionades, aquest treball no hauria estat possible.

Resum

Aquest treball explora la viabilitat de desenvolupar un sistema de classificació per a assaigs experimentals en l'àmbit de la dinàmica del vehicle. Per representar els assaigs es construeixen variables sintètiques, s'avalua la seva rellevància i es compara un model supervisat (Random Forest) amb un de no supervisat (Isolation Forest) per determinar quin enfocament d'entrenament és més adient. Constatada la millor adaptació dels models supervisats, s'entrenen diversos algoritmes supervisats de funcionament diferent per estudiar si la limitació rau en la natura del model o en les dades. L'estudi revela problemes de sobreajustament, soroll en les etiquetes i manca de dades. Com a resposta, es desenvolupa una eina gràfica per millorar el procés d'etiquetatge i facilitar futures millores en els entrenaments.

Paraules clau: Assaig, canals de dades, gràfiques, *Machine Learning*, característiques, variables sintètiques, entrenament supervisat, *Random Forest, XGBoost, Isolation Forest*, mètriques d'avaluació, exactitud, sensibilitat, precisió, *F1-Score*, sobreajustament, validació creuada.

Resumen

Este trabajo explora la viabilidad de desarrollar un sistema de clasificación para ensayos experimentales. Para representar los ensayos se construyen variables sintéticas, se evalúa su relevancia y se compara un modelo supervisado (Random Forest) con uno no supervisado (Isolation Forest) para determinar qué enfoque de entrenamiento es más adecuado. Confirmada la mejor adaptación de los modelos supervisados, se entrenan varios algoritmos supervisados de funcionamiento distinto para estudiar si la limitación proviene de la naturaleza del modelo o de los datos. El estudio revela problemas de sobreajuste, ruido en las etiquetas y escasez de datos. Como respuesta, se desarrolla una herramienta gráfica para mejorar el proceso de etiquetado y facilitar futuras mejoras en los entrenamientos.

Palabras clave: Ensayo, canales de datos, gráficas, *Machine Learning*, características, variables sintéticas, entrenamiento supervisado, *Random Forest, XGBoost, Isolation Forest*, métricas de evaluación, exactitud, sensibilidad, precisión, *F1-Score*, sobreajuste, validación cruzada.

Abstract

This work explores the feasibility of developing a classification system for experimental test data. Synthetic variables are constructed to represent the tests, their relevance is evaluated, and a supervised model (Random Forest) is compared with an unsupervised one (Isolation Forest) to determine the most suitable training approach. Once the superiority of supervised models is confirmed, several supervised algorithms with different mechanisms are trained to assess whether the limitations stem from the model itself or the data. The study reveals issues such as *overfitting*, noisy labels, and lack of data. As a solution, a graphical tool is developed to enhance the labeling process and support future improvements in model training.

Keywords: Experimental test, data channels, plots, *Machine Learning*, features, synthetic variables, supervised training, *Random Forest, XGBoost, Isolation Forest*, evaluation metrics, accuracy, recall, precision, *F1-Score*, overfitting, cross-validation.

Índex

Índex de figures

Índex de taules

1 Introducció

1.1 Applus+ IDIADA i el Departament de *Vehicle Dynamics*

Applus+ IDIADA [1] és una empresa líder en el sector de l'automoció. Actualment supera el nombre de 3.400 treballadors distribuïts en més de 24 països i té la seva seu central a Santa Oliva, Tarragona, lloc on disposa de les pistes de proves més avançades d'Europa. L'empresa ofereix serveis d'enginyeria, proves, disseny i homologació per ajudar fabricants i proveïdors a millorar el rendiment, la seguretat i l'eficiència dels seus vehicles així com a la comparació i desenvolupament de prototips.

És en aquesta seu d'IDIADA on es va desenvolupar el treball de fi de grau.

La dinàmica del vehicle és la branca de l'enginyeria de l'automoció que estudia el comportament dels vehicles en moviment, analitzant com responen a diferents forces i condicions de conducció. El departament de dinàmica s'encarrega de l'estudi de la suspensió, la direcció, els pneumàtics, l'estabilitat i el control del vehicle, amb l'objectiu de millorar la seguretat, el confort i el rendiment dinàmic.
Es treballa en aspectes com la distribució de masses, la transferència de càrrega, l'adhesió dels pneumàtics a la carretera i la interacció entre el xassís i el sistema de suspensió. Mitjançant simulacions, assajos físics i models matemàtics, s'analitza el comportament del vehicle i es comparteixen les observacions i resultats amb el client per tal de que aquest pugui optimitzar el disseny del vehicle per garantir una resposta òptima en diferents situacions de conducció: corbes, frenades o acceleracions, contribuint així a l'eficiència i seguretat global del vehicle.

1.2 *Kinematics and Compliance (K&C)*

El subdepartament de *Kinematics and Compliance* s'ocupa de l'estudi del comportament de la suspensió i la direcció d'un vehicle mitjançant assaigs quasi estàtics. Aquestes proves permeten caracteritzar com es desplacen les rodes respecte al xassís sota diferents condicions de càrrega i moviment.

Les proves *Kinematics & Compliance (K&C)* són assajos experimentals que s'utilitzen per avaluar el comportament de la suspensió d'un vehicle en resposta a diferents forces i moviments aplicats. Aquestes proves són essencials per comprendre com varien els paràmetres geomètrics i mecànics de la suspensió en diferents condicions de càrrega i moviment. Aquests assaigs es divideixen en dos tipus principals: els assajos de cinemàtica (*Kinematics*) i els assajos de compliança (*Compliance*).

Assajos *Kinematics*

Els assajos de cinemàtica estudien com es mou la suspensió quan es produeixen canvis en la posició de les rodes. Aquests assajos permeten determinar la geometria real del sistema de suspensió i com aquesta influeix en la dinàmica del vehicle.

- ***Vertical Motion*:** En aquest tipus d'assaig, les plataformes del banc de proves es mouen verticalment amunt i avall, simulant el recorregut de la suspensió quan el vehicle passa per irregularitats en la carretera o experimenta càrregues verticals. L'objectiu és analitzar com varien la convergència, la caiguda i la batalla en funció del recorregut de la suspensió.
- ***Roll Motion*:** Aquest assaig simula el moviment de rol (*roll*) del vehicle, on la carrosseria s'inclina lateralment com a conseqüència d'una força centrífuga, per exemple, durant un canvi de direcció a alta velocitat. Es busca determinar com afecta aquest moviment a la geometria de la suspensió i a la resposta del vehicle en termes de transferència de càrrega lateral i estabilitat.

Assajos *Compliance*

Per altra banda, els assajos de compliança se centren en l'elasticitat dels components de la suspensió quan es sotmeten a forces externes, avaluant deformacions en elements com els braços de suspensió, les articulacions i els suports de muntatge. Aquestes proves són clau per entendre com la flexibilitat estructural afecta la resposta del vehicle davant forces laterals, longitudinals i de diferents moments de força aplicats a la roda, així com per identificar possibles desviacions no desitjades en el comportament de la suspensió.

- ***Steering Compliance*:** En aquest assaig s'apliquen forces al sistema de direcció per simular les càrregues generades durant la conducció i analitzar com els components elàstics afecten la precisió i la resposta del vehicle en girs. Es busca avaluar si hi ha desviacions o retard en la resposta de la direcció degut a la flexibilitat dels components.
- ***Longitudinal Compliance*:** Aquest assaig consisteix en aplicar forces en la direcció longitudinal del vehicle per simular les forces de frenada i acceleració. L'objectiu és determinar com es deformen els components de la suspensió davant aquestes forces i com poden influir en l'estabilitat i el control del vehicle.
- ***Lateral Compliance*:** En aquest cas, s'apliquen forces laterals a les rodes per simular les càrregues generades durant una corba. Es busca entendre com els components elàstics de la suspensió permeten petits desplaçaments laterals de les rodes i com això pot afectar la resposta i el comportament del vehicle en situacions de canvi de trajectòria.
- ***Aligning Torque Compliance*:** Aquest assaig mesura la resposta de la suspensió davant torques d'alineació aplicades a les rodes, que es produeixen naturalment quan el pneumàtic genera forces de gir. Es busca determinar com es transmet aquest torque a través dels components de la suspensió i com pot afectar la sensació de direcció i estabilitat del vehicle.

Metodologia Operativa del Subdepartament

El subdepartament de *Kinematics and Compliance* segueix una metodologia clara i estructurada per a la realització i validació dels assaigs. Aquest procés es pot dividir en diverses fases:

1. **Planificació amb el client:** Inicialment, es manté una reunió entre el client i els enginyers responsables per definir els objectius del projecte, determinar quins assaigs es duran a terme i establir les especificacions concretes per a cadascun d'ells (com ara la velocitat, càrrega, altura de l'assaig, etc.).
2. **Instrumentació del vehicle:** Un cop definits els assaigs, es procedeix a la preparació del vehicle. Això inclou la instrumentació necessària per registrar les variables físiques rellevants i assegurar que el cotxe estigui llest per ser col·locat al banc de proves.
3. **Execució dels assaigs:** Cada projecte pot incloure múltiples assaigs, que poden ser del mateix tipus (per exemple, diversos assaigs *Vertical Motion*) però amb condicions diferents. L'equip realitza els assaigs al banc de proves seguint rigorosament les condicions establertes.
4. **Generació d'arxius de dades:** En finalitzar cada assaig, el banc de proves transmet les dades recollides a un ordinador connectat, generant un fitxer de dades únic per a cada assaig. Aquest fitxer conté totes les mesures enregistrades durant l'assaig i esdevé la base per a la validació. El format d'aquest fitxer es comú per a tots els assaigs.
5. **Validació de l'assaig:** L'enginyer responsable duu a terme una validació tècnica de cada assaig, que es divideix en dues fases:

- **Comprovació de paràmetres:** Es verifica que els paràmetres configurats abans de començar l'assaig coincideixen amb les especificacions pactades amb el client.
- **Anàlisi de gràfiques:** S'inspeccionen les gràfiques generades per detectar comportaments no desitjats (com soroll excessiu, pics inusuals, formes anòmales, etc.). El tipus de gràfiques analitzades depèn del tipus d'assaig realitzat.

6. **Repetició si cal:** Si en alguna de les fases de validació es detecta una incidència, l'assaig pot ser repetit per garantir la qualitat i fiabilitat del conjunt de dades del projecte.

Aquesta metodologia assegura que les dades obtingudes siguin representatives i fiables, la qual cosa és fonamental per al correcte desenvolupament d'estudis posteriors, com els analitzats en aquest treball.

1.3 Definició del problema i objectius

1.3.1 Context i Motivació

En el marc de l'enginyeria del vehicle, l'anàlisi i validació de dades experimentals obtingudes d'assaigs és fonamental per garantir la qualitat i el rendiment dels components del vehicle assajat. En particular, fer una avaluació correcta del comportament del vehicle en assaigs de laboratori permet detectar possibles anomalies o comportaments no desitjats, els quals podrien comprometre la seguretat o la durabilitat del producte final.
Fins a dia d'avui, aquesta tasca de validació ha estat portada a terme per enginyers experts que, basant-se en la seva experiència i en l'anàlisi visual de múltiples gràfiques, determinen si un assaig és vàlid o presenta problemes que requereixen correcció. Aquest procés, tot i ser molt fiable, és altament dependent de la subjectivitat de l'observador, requereix un temps considerable i es veu limitat per la càrrega de treball humana.
L'objectiu general d'aquest projecte és explorar si és possible el desenvolupament d'un sistema automàtic, basat en tècniques d'aprenentatge automàtic (*Machine Learning*), capaç de simular amb la màxima precisió les decisions que actualment prenen els enginyers experts a l'hora de decidir si un assaig es vàlid o no. Aquest sistema, de ser viable, podria agilitzar els processos de validació, reduir errors humans i alliberar recursos per a tasques de major valor afegit.

1.3.2 Àmbit del Treball

Aquest Treball de Fi de Grau es desenvolupa en col·laboració amb l'empresa **Applus+ IDIADA**, reconeguda multinacional en el sector de l'automoció, especialment en serveis d'enginyeria, assaigs i certificació de vehicles.

El projecte s'emmarca dins l'àrea de *Chassis, Active Safety and Durability*, concretament en el departament de *Vehicle Dynamics*, i més específicament dins el subdepartament de *Kinematics and Compliance (KnC)*.

1.3.3 Descripció del Problema

Actualment, la validació dels assaigs del subdepartament de K&C es realitza mitjançant l'anàlisi manual de múltiples gràfiques generades durant la prova de cada vehicle. Cada assaig inclou diverses gràfiques (canals de dades), cadascuna mesurant diferents paràmetres (força, desplaçament, angles, moments, etc.).

Aquest procés de validació es basa en criteris que combinen el coneixement tècnic i l'experiència. No obstant això, presenta limitacions importants:

- Dependència d'experts humans, amb la subjectivitat que això implica.
- Consum de temps considerable per revisar manualment cada gràfica significativa de cada assaig.
- Possibilitat d'errors o inconsistències degudes a la fatiga o al volum de treball.
- Comportaments anòmals en productes prototips.

Aquests són els principals obstacles que el model d'aprenentatge automàtic haurà d'afrontar. A causa d'aquests factors, és previsible que la precisió final del model no sigui excel·lent, ja que el soroll inherent en les dades i la complexitat dels assaigs dificulten una classificació perfecta.

A més, el sistema d'etiquetatge es relativament nou en el subdepartament. El fet de que en un projecte estiguin separats els arxius de dades bons dels arxius de dades dolents només es troba present en els projectes dels darrers 4 anys, cosa que provocarà que les dades útils per a models d'aprenentatge automàtic basats en etiquetes de classificació (supervisats) siguin inferiors a les dades útils per a enfocaments que no requereixen etiquetes explícites (no supervisats). D'altra banda, les etiquetes actuals dels

assaigs només són fiables a nivell assaig i no a nivell de gràfica individual, en altres paraules, se sap que un assaig ha sortit malament però no sabem quin ha estat el conjunt de dades (gràfica) culpable, afegint complexitat al problema.

1.3.4 Objectius específics del Treball

Aquest projecte té com a objectiu principal determinar si es possible el desenvolupament d'un sistema de classificació d'assaigs de cinemàtica i conformitat (K&C), robust i eficient a partir de dades extretes de gràfiques d'assaigs experimentals. Per assolir aquest objectiu general, s'han definit els següents objectius específics:

- **Aplicar tècniques de selecció de característiques**: Identificar i seleccionar les variables més representatives per a cada conjunt de gràfiques, amb l'objectiu de maximitzar la rellevància de la informació i reduir el soroll o redundància present a les dades originals.
- **Justificar la generació de variables sintètiques**: Construir característiques agregades (mitjanes, desviacions estàndard, valors màxims, mínims, etc.) que sintetitzin la informació temporal de les gràfiques i justifiquin el seu ús com a representació a nivell d'assaig.
- **Determinar el mètode d'entrenament més adient**: Comparar models entrenats amb tècniques supervisades i no supervisades per determinar quin enfocament és més adequat per al problema plantejat, tenint en compte la naturalesa de les dades i la disponibilitat de mostres etiquetades.
- **Comparar l'eficàcia entre models supervisats i no supervisats**: Avaluar el rendiment de models completament supervisats en comparació amb models no supervisats, que posteriorment són ajustats amb dades parcialment etiquetades, tot analitzant el grau d'informació necessària per assolir una classificació precisa.
- **Seleccionar models del tipus d'entrenament escollit i comparar-ne el rendiment**: Un cop identificat el tipus d'entrenament més adequat (supervisat o no supervisat), explorar i comparar diferents models dins d'aquest enfocament (*Random Forest, XGBoost, KNN*, etc.) per escollir el més òptim.
- **Avaluar el rendiment dels models**: Utilitzar mètriques com l'exactitud, *F1-Score*, precisió i sensibilitat, matrius de confusió i anàlisis qualitatives per valorar la capacitat predictiva dels models i la seva robustesa.
- **Identificar el millor model i determinar el seu marge de millora**: Un cop seleccionat el millor model, analitzar si existeix marge de millora (mitjançant més dades, millor selecció de característiques, optimització d'hiperparàmetres, etc.) i definir els passos futurs per continuar incrementant el rendiment del sistema.
- **Desenvolupament d'una eina d'utilitat pràctica**: Tant si s'assoleixen resultats competitius amb algun dels models d'aprenentatge automàtic com si no, es desenvoluparà una eina que aporti valor afegit al procés. En cas d'obtenir un model fiable, l'eina podrà incorporar-lo com a sistema automàtic de classificació d'assaigs. En cas contrari, l'eina servirà com a instrument de suport al procés d'etiquetatge manual, permetent construir, de manera més estructurada i eficient, conjunts de dades supervisades que facilitin futurs entrenaments.

2 Software i mètodes

2.1 Software

Durant el desenvolupament d'aquest treball s'ha fet ús de diverses eines de programari i Llibreries de codi obert, amb l'objectiu de processar dades, construir models de *machine learning*, analitzar resultats i desenvolupar eines interactives.

2.1.1 Entorn de Desenvolupament

- **Visual Studio Code**[16]: Entorn emprats per a l'edició del codi i implementació de *Python* i *Jupyter Notebook*.
- **Python** [5]: Ha estat el llenguatge de programació utilitzat. Es tracta d'un llenguatge altament llegible, versàtil i àmpliament adoptat en la comunitat científica i d'anàlisi de dades.
- **Jupyter Notebook**: S'ha emprat com a estructura interactiva dins l'etorn de *Visual Studio Code* per a l'execució de codi Python, visualització de resultats i documentació simultània. Aquesta eina ha permès realitzar proves ràpides i tenir una traçabilitat clara del procés d'experimentació.

2.1.2 Llibreries de Python

- **pandas** [4]: Utilitzada per a la manipulació i anàlisi de dades en forma de taules. Ha estat clau per llegir, netejar, fusionar i estructurar conjunts de dades provinents de fitxers `.csv` o `.tab`.
- **numpy** [3]: Ha proporcionat funcionalitats per a càlculs numèrics i manipulació eficient de vectors i matrius. Molt útil per calcular estadístiques com mitjanes, desviacions, *skewness*, *kurtosis*, entre altres.
- **matplotlib** [2] i **seaborn** [7]: Aquestes Llibreries han estat fonamentals per a la generació de gràfiques, incloent visualitzacions exploratòries, corbes ROC, matrius de confusió, comparacions de mètriques i importància de característiques. *Seaborn* ofereix un estil més polit i integració amb *pandas*.
- **scikit-learn (sklearn)** [6]: Biblioteca central per a la construcció i avaluació dels models de *machine learning*.
- **os** i **re**: Llibreries de gestió de fitxers i expressions regulars, emprades principalment en l'eina gràfica per llegir i organitzar fitxers d'assaig i identificar noms rellevants.
- **tkinter** [12], **matplotlib.backends**, **watchdog** [14], **xlwings**: Aquestes Llibreries han estat clau per al desenvolupament de la interfície gràfica d'etiquetatge d'errors. Han permès detectar nous fitxers, llegir-ne el contingut, mostrar gràfiques interactives i generar fitxers `.csv` amb les seleccions fetes pels usuaris.

En conjunt, aquesta combinació de programari ha permès desenvolupar un flux de treball robust, flexible i completament documentable, des del pre-processament fins a l'anàlisi de resultats i la generació d'una eina funcional per a l'etiquetatge manual.

2.2 Mètodes

L'aprenentatge automàtic és una branca de la intel·ligència artificial que permet als ordinadors aprendre patrons a partir de dades i prendre decisions o fer prediccions sense ser explícitament programats.

L'aprenentatge supervisat implica entrenar un model amb dades etiquetades, és a dir, se sap si les dades pertanyen a un grup C_1, C_2 o C_n en el cas de que es tinguin n classes diferents. Alguns exemples de models supervisats són regressions, arbres de decisió, xarxes neuronals, etc.
L'aprenentatge no supervisat implica treballar amb dades no etiquetades. L'objectiu és trobar patrons ocults, agrupacions o anomalies dins les dades. Exemples comuns són *K-means, Isolation Forest*, etc.

2.2.1 Descripció de les Característiques

En el context d'aprenentatge automàtic, les **característiques** (o *features*) són les variables que descriuen cada instància o observació dins d'un conjunt de dades. L'elecció i l'extracció correcta de les característiques són fonamentals per garantir un bon rendiment del model. A continuació, es detallen algunes mesures estadístiques clau utilitzades per descriure dades numèriques:

Mitjana

La **mitjana** és una mesura de tendència central que representa el valor promig d'un conjunt de dades. Es calcula com la suma de tots els valors dividida pel nombre total de valors:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Desviació estàndard

La **desviació estàndard** mesura la dispersió o variabilitat dels valors respecte a la mitjana. Una desviació estàndard elevada indica que els valors estan molt dispersos, mentre que una baixa indica que estan propers a la mitjana:

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2}$$

Curtosi

La **curtosi** mesura com de punxeguda és una distribució. Una curtosi elevada indica una distribució amb pics molt marcats i cues pesades, mentre que una baixa indica una distribució més plana:

$$\text{Curtosi} = \frac{\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^4}{\left(\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right)^2}$$

Asimetria

L'**asimetria** mesura la falta de simetria d'una distribució. Una asimetria positiva indica que la cua dreta és més llarga; una asimetria negativa indica que la cua esquerra és més llarga:

$$\text{Asimetria} = \frac{\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^3}{\sigma^3}$$

Altres característiques comunes poden incloure els valors màxims, mínims o fins i tot les mateixes mesures comentades però aplicades a la derivada de la funció.

2.2.2 Mètriques d'Avaluació de classificació

Per tal de quantificar el rendiment dels models de classificació, es fan servir diferents mètriques que tenen en compte tant la precisió com la cobertura del model.

Matriu de Confusió

La matriu de confusió és una eina fonamental en l'avaluació del rendiment dels models de classificació. Permet analitzar no només l'eficàcia general del model, sinó també detallar on exactament s'estan cometent errors. Aquesta matriu organitza les prediccions en quatre categories:

- ***True Positive* (TP)**: Nombre de casos en què el model ha predit correctament la classe positiva (Classse 1), és a dir la classe objectiu, en el nostre cas la classe objectiu seran els arxius de dades no vàlids.
- ***True Negative* (TN)**: Nombre de casos en què el model ha predit correctament la classe negativa (Classe 0).
- ***False Positive* (FP)**: Nombre de casos en què el model ha predit com a positiva una instància que en realitat era negativa (també conegut com a error de tipus I).
- ***False Negative* (FN)**: Nombre de casos en què el model ha predit com a negativa una instància que realment era positiva (també conegut com a error de tipus II).

Aquests valors es poden representar en forma de taula:

	Predicció Negativa (Classe 0)	**Predicció Positiva (Classe 1)**
Real Negatiu (Classe 0)	TN	FP
Real Positiu (Classe 1)	FN	TP

Taula 1: Matriu de Confusió

En el nostre cas, la classe 1 correspon als assaigs erronis i la classe 0 als assaigs bons.
A partir de la matriu de confusió, es poden calcular diverses mètriques importants:

- **Exactitud (Accuracy)**: proporció de prediccions correctes respecte al total d'instàncies. És una mètrica global, però pot resultar enganyosa en casos de classes desequilibrades, ja que no distingeix entre tipus d'errors.
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
- **Precisió (*Positive Prediction Value*)**: mesura la qualitat de les prediccions positives. Indica quin percentatge de les instàncies classificades com a positives ho són realment. És especialment rellevant quan el cost dels falsos positius és elevat.
$$Precision = \frac{TP}{TP + FP}$$
- **Sensibilitat (*True Positive Rate*)**: La sensibilitat o *recall*, mesura la capacitat del model per detectar les instàncies positives. Indica quin percentatge dels positius reals ha estat capturat pel model. És fonamental quan el cost dels falsos negatius és alt.
$$Recall = TPR = \frac{TP}{TP + FN}$$

- **Taxa de falsos positius (False Positive Rate)**: La taxa de falsos positius o *fall-out*, indica la proporció de negatius reals que han estat incorrectament classificats com a positius. És útil per entendre la quantitat d'errors del model sobre la classe negativa, i és un component fonamental de la corba ROC.

$$Fall\text{-}out = FPR = \frac{FP}{FP + TN}$$

- **F1-Score**: és la mitjana harmònica entre la precisió i la sensibilitat. Aquesta mètrica proporciona un equilibri entre ambdues i és especialment útil en escenaris amb desbalanceig de classes, ja que penalitza tant els falsos positius com els falsos negatius.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Corba ROC i AUC

Una altra eina important per avaluar el rendiment d'un model de classificació binària és la **corba ROC** (*Receiver Operating Characteristic*). Aquesta corba representa la relació entre la **sensibilitat** (*True Positive Rate*, TPR) i la **taxa de falsos positius** (*False Positive Rate*, FPR) per a diferents llindars de classificació.

- L'eix vertical mostra la **sensibilitat (TPR)**, és a dir, la proporció de positius correctament identificats pel model, també anomenada ***Recall***.
- L'eix horitzontal mostra la **taxa de falsos positius (FPR) o *fall-out***, la proporció de negatius incorrectament classificats com a positius.

En la següent Figura (1), es poden observar diversos tipus de classificadors:

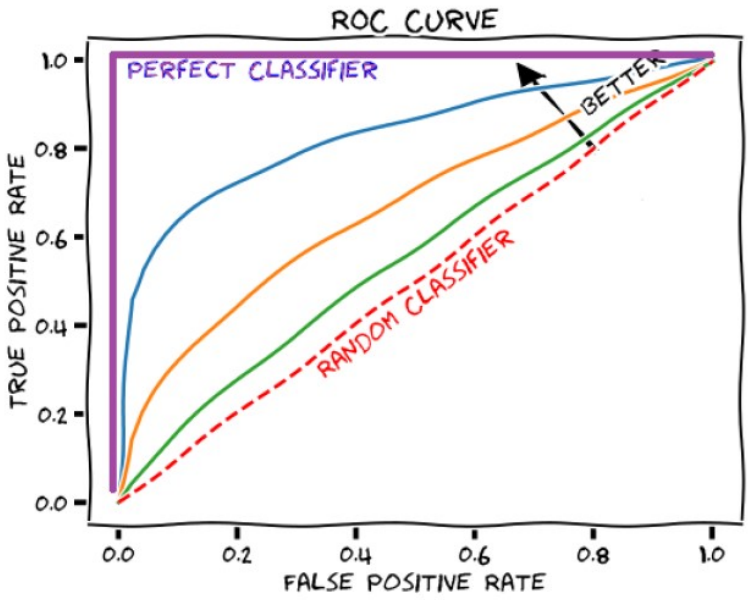

Figura 1: Exemple de corba ROC per diferents classificadors.

- El **classificador perfecte** (línia morada) aconsegueix una TPR de 1 amb FPR de 0, és a dir, una classificació ideal.
- La línia vermella discontínua representa un **classificador aleatori**, que no té cap capacitat predictiva (AUC = 0,5).
- Les corbes en blau, taronja i verd mostren models amb rendiment progressivament inferior.

La qualitat global d'un model es pot quantificar mitjançant l'**AUC** (*Area Under the Curve*). Aquest valor representa l'àrea sota la corba ROC:

- Un valor d'**AUC = 1.0** indica un model perfecte.
- Un valor d'**AUC = 0.5** correspon a un model aleatori.
- Com més proper és l'AUC a 1, millor és la capacitat predictiva del model.

Aquesta mètrica és especialment útil en situacions de desequilibri de classes, ja que no depèn d'un únic llindar de classificació sinó que considera tots els possibles.

2.2.3 Sobreajustament (Overfitting)

El sobreajustament és un fenomen comú en l'aprenentatge automàtic que es produeix quan un model aprèn massa bé el conjunt de dades d'entrenament, capturant tant els patrons rellevants com el soroll o les anomalies específiques d'aquell conjunt concret. Això pot donar lloc a un rendiment aparentment excel·lent durant l'entrenament però a una capacitat molt limitada per generalitzar el coneixement a noves dades no vistes, com les del conjunt de validació o de prova.

Des del punt de vista tècnic, el sobreajustament apareix quan el model té massa capacitat o complexitat en relació amb la quantitat i la qualitat de les dades disponibles. Alguns símptomes típics són: una precisió molt alta en l'entrenament i una precisió baixa en validació o test, prediccions que canvien de manera exagerada per petites variacions en les dades d'entrada.

Aquest problema es veu agreujat en situacions com:

- Conjunts de dades petits o amb molt soroll.
- Etiquetatge inconsistent o subjectiu.
- Models massa flexibles (com arbres de decisió profunds, xarxes neuronals grans...).
- Absència de tècniques de regularització o validació adequada.

Per evitar o minimitzar el sobreajustament, es poden aplicar diverses estratègies:

- **Regularització**: Són tècniques que penalitzen la complexitat del model per evitar ajustaments excessius. Per exemple, la regularització L1 afavoreix solucions esparses eliminant característiques menys rellevants, mentre que la L2 redueix la magnitud dels pesos sense anul·lar-los. En el cas dels arbres de decisió, es pot aplicar pruning (poda), que elimina branques poc informatives per evitar un excés d'ajust.
- **Validació creuada (*Cross-validation*)**: Consisteix a dividir les dades en múltiples subconjunts per entrenar i validar iterativament, proporcionant una estimació més estable del rendiment real del model.
- **Reducció de la complexitat del model**: Com per exemple limitar la profunditat d'un arbre de decisió o reduir el nombre de neurones en una xarxa neuronal, per evitar que el model pugui memoritzar el conjunt d'entrenament.
- **Augment de dades**: S'utilitza per generar noves mostres artificialment, augmentant la diversitat del conjunt d'entrenament i ajudant a millorar la generalització, especialment en contextos amb poques dades.
- **Aturada anticipada**: És una tècnica que interromp l'entrenament d'un model (normalment en xarxes neuronals) quan es detecta que el rendiment sobre el conjunt de validació comença a empitjorar, evitant que el model continuï ajustant-se de manera excessiva a les dades d'entrenament.

En conclusió, el sobreajustament és un dels principals riscos en la construcció de models predictius, ja que pot conduir a conclusions enganyoses respecte a la seva eficàcia. És especialment rellevant en contextos amb dades limitades, com el cas d'aquest estudi, on cal tenir molta cura a l'hora d'avaluar la veritable capacitat de generalització d'un model.

3 Mineria de dades i selecció de característiques

3.1 Mineria de dades

La primera fase del projecte ha consistit en la recopilació, filtratge i lectura de les dades experimentals provinents dels diferents projectes, recordar que un projecte està format per varis assaigs de diferents tipus i que cadascun d'aquests conté el seu arxiu de dades respectiu. Aquest procés s'ha automatitzat mitjançant diversos scripts en Python, els quals es trobaran a l'Annex.

3.1.1 Format de les dades

Els arxius de dades es poden identificar fàcilment, ja que son els únics arxius dins de les carpetes de projectes amb extensió *.tab*, aquesta extensió indica simplement que es tracta d'un arxiu de text on les dades estan separades per tabulacions.
Per a la lectura d'aquests arxius s'ha de tenir en compte la seva estructura, aquesta es pot desglossar en dos sub-estructures. La primera conté sobretot dades pre-assaig, per exemple: l'eix del vehicle que s'està assajant, l'ample de via, etc. Aquesta informació ve donada per dues columnes, una amb el nom del paràmetre i l'altre amb al valor. Aquest format s'estén unes 180 línies.
Seguidament es troben les dades recollides a l'assaig i el document pren una altra sub-estructura, aquesta conté aproximadament unes 100 columnes d'uns 1800 valors cadascuna (depenent de la llargada de l'assaig), cadascuna d'aquestes columnes correspon a canals independents de gravació de dades del banc de proves: hi ha un canal pel temps, un altre per la posició vertical de la plataforma esquerra, una per la plataforma dreta, un altre per la força vertical, etc.

A continuació es mostra la part del document de dades on es fa el canvi entre el primer sub-conjunt de dades i el segon que correspon a les dades gravades a l'assaig.

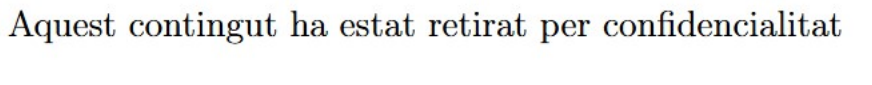

Figura 2: Primera i segona sub-estructura del document de dades

Per aquest treball només es treballarà amb certs canals (depenent del tipus d'assaig) de la segona part de les dades, segona sub-estructura del document. Gràcies a aquests canals s'obtindran les gràfiques pertinents a analitzar de l'assaig.

3.1.2 Ubicació de les Dades Originals

Cada projecte té una carpeta única i dins d'aquesta es troba tota la documentació pertinent. Les dades de cada projecte, s'ubicaven en carpetes anomenades *Raw Data*. Tanmateix, en projectes més antics, aquestes dades podien estar dins de carpetes anomenades *Finals*, en el cas de que l'arxiu *Raw Data* es trobés vuit. En aquest últim cas, es considerava que tots els arxius eren correctes (o "bons") ja que s'assumia que havien passat un primer filtre de validació, d'altre banda, els arxius que es trobessin en la carpeta *Raw Data*, s'hauria de mirar en següents passos si existia, o no, classificació d'arxius de dades bons i dolents.

3.1.3 Procés de Recollida de les Dades

Un cop identificades les possibles ubicacions de les dades dins l'estructura de carpetes dels projectes, es va desenvolupar un script en Python per automatitzar el procés de recopilació. L'objectiu principal era

localitzar carpetes que continguessin dades crues, habitualment anomenades *Raw Data*, i en alguns casos *Finals*, i copiar-ne el contingut a una ubicació comuna per al seu posterior processament.

Aquest procés es va implementar utilitzant diverses llibreries del llenguatge Python:

- `os`: per recórrer de manera recursiva l'estructura de carpetes i accedir al contingut dels directoris.
- `shutil`: per copiar carpetes i arxius seleccionats a la destinació corresponent.
- `pathlib`: per gestionar rutes de fitxers amb una sintaxi moderna i robusta.
- `time`: considerada per a funcionalitats relacionades amb temporització, tot i que finalment no s'ha utilitzat.

El codi funciona recorrent sistemàticament l'arbre de directoris que conté els projectes, i identifica carpetes amb noms que contenen la cadena `Raw Data`. En cas que aquestes no continguin dades, també es consideren carpetes anomenades `Finals`, que poden contenir els arxius ja filtrats i validats. Els continguts d'aquestes carpetes es copien a una ubicació central de treball, generant noms de carpeta simplificats per evitar errors relacionats amb rutes massa llargues, i afegint un identificador incremental per evitar sobreescriptures.

Durant la fase de còpia, es descarten automàticament tots aquells arxius que no són rellevants per al processament de dades tècniques, com ara els que tenen extensions `.pdf`, `.emf`, `.jpg`, `.dat`, entre d'altres. Aquesta selecció es fa mitjançant una funció de filtratge definida específicament per a aquest propòsit.

Per assegurar que només es treballi amb informació tècnica útil, l'script també exclou carpetes amb noms habituals associats a documentació, fotografies o informes (ex: `documentation`, `photos`, `report`, `film`, etc.). Un cop completada la còpia, s'elimina qualsevol carpeta buida que hagi quedat a la destinació.

Aquest procés automatitzat ha permès optimitzar significativament la fase de recopilació i preselecció de dades, evitant errors manuals i reduint el temps necessari per a preparar grans volums d'informació experimental.

3.1.4 Identificació de Projectes Classificats i Separació de dades

Per tal de poder utilitzar un enfocament de classificació supervisada, es va buscar, mitjançant un altre codi, quins projectes contenien carpetes que indiquessin explícitament si les dades eren bones o dolentes, és a dir, quins projectes comptaven amb classificació d'assaigs. S'identificaren noms de carpeta com *NO*, *Bad*, *Discarded*, *Not Selected*, etc. Separant d'aquesta forma totes les dades útils per al model supervisat de les que no.

Un cop identificats tots els projectes amb classificació, es van separar, gràcies a un codi de *python*, tots els arxius de dades donats com a bons dels dolents.

3.1.5 Classificació per Tipus d'Assaig

Un feta la separació de les dades, aquestes es van agrupar segons els sis tipus d'assaigs disponibles: *Vertical, Roll, Steering, Longitudinal, Lateral i Aligning Torque.*

Per a cada tipus d'assaig es va generar una carpeta pròpia, diferenciant les dades bones de les dolentes, permetent així un anàlisi posterior per modelatge i classificació més eficient. Aquesta classificació es va fer tant pel conjunt de dades classificades com pel conjunt de dades sense classificar, òbviament mantenint la distinció de dades classificades i no classificades.

Un cop acabada aquesta classificació, totes les dades ja es troben llestes per a la seva anàlisi i *feature engeneering.*

3.2 Selecció de característiques

La selecció de característiques o *feature selection* és el procés en el qual s'intenta descriure de manera òptima un conjunt extens de dades en un nombre menor de dades representatives o noves variables. Aquestes noves variables o característiques faciliten als algoritmes d'aprenentatge automàtic millorar el seu rendiment i capacitat predictiva, sempre i quan les noves variables siguin un bon representant de les dades inicials.
Aquest procés inclou:

- **Selecció de característiques:** Tria de les variables més rellevants dins del conjunt de dades original, descartant aquelles que són irrellevants o redundants.
- **Creació de noves característiques:** Generació de noves variables a partir de les existents, com per exemple transformacions matemàtiques, agregacions, o extracció d'estadístiques.
- **Normalització i estandardització:** Ajust de l'escala de les dades per facilitar l'aprenentatge dels models, sempre i quan sigui necessari o convenient.

3.2.1 Justificació de l'enfocament basat en enginyeria de característiques

Inicialment, es va considerar la possibilitat de desenvolupar un model de *Machine Learning* més complex. L'objectiu era que el model identifiqués, donat un arxiu de dades nou, si aquest era correcte o incorrecte, basant-se en si tots els conjunts de dades (gràfiques de l'assaig) a mirar es donaven com a vàlids o hi havia algun que no. Cada document conté múltiples gràfiques (conjunts de dades) a revisar depenent del tipus d'assaig que s'analitzi, i es considera dolent si una sola gràfica presenta un comportament no desitjat.
Com ja s'ha comentat prèviament, durant el procés de mineria de dades, el nombre de dades amb una classificació fiable foren menys de les esperades. Aquests fets van provocar una reducció dràstica en el volum de dades aprofitables per a un model supervisat.
Per tractar de compensar el nombre de dades obtingudes (al voltant de 1000 arxius per cada tipus d'assaig, 1000x6) es va optar per un enfocament alternatiu: Selecció de característiques (*feature selection*). Aquest mètode permet:

- Treballar amb un nombre menor de dades etiquetades.
- Centrar-se en patrons específics detectables gràcies a la morfologia de les gràfiques.
- Desenvolupar un model més senzill però robust, orientat a la detecció de comportaments anòmals en les diferents formes de senyals.

En aquest treball, s'ha optat per fer ús d'aquest mètode per compensar la manca de volum de dades, centrant l'atenció en l'extracció de característiques sintètiques rellevants a partir de gràfiques experimentals generades durant els assaigs.

3.2.2 Classificació de les Gràfiques

Amb ajuda dels experts de l'empresa i mitjançant anàlisi visual es va decidir categoritzar les gràfiques dels assaigs en tres categories:

1. Gràfiques que segueixen un patró clar, com podrien ser comportaments sinusoidals o semblants.
2. Gràfiques les quals es desitja que el seu contingut estigui al voltant del 0.
3. Finalment les gràfiques especifiques de cada assaig o gràfiques amb comportaments més complexes. Cal concretar que a diferència de les anteriors aquestes no son sèries temporals, pel que l'anàlisi i descripció de les dades és més delicat.

Aquesta classificació es va crear per poder descriure amb diferents característiques els diferents grups de gràfiques. En els dos primers grups, les variables sintètiques encarregades de descriure el comportament de les dades seran les mateixes per a tots els tipus d'assaigs. En canvi, les gràfiques específiques d'assaig, variaran segons l'assaig que s'estigui tractant.

A continuació es mostren exemples de cadascun dels tipus de gràfiques mencionats.

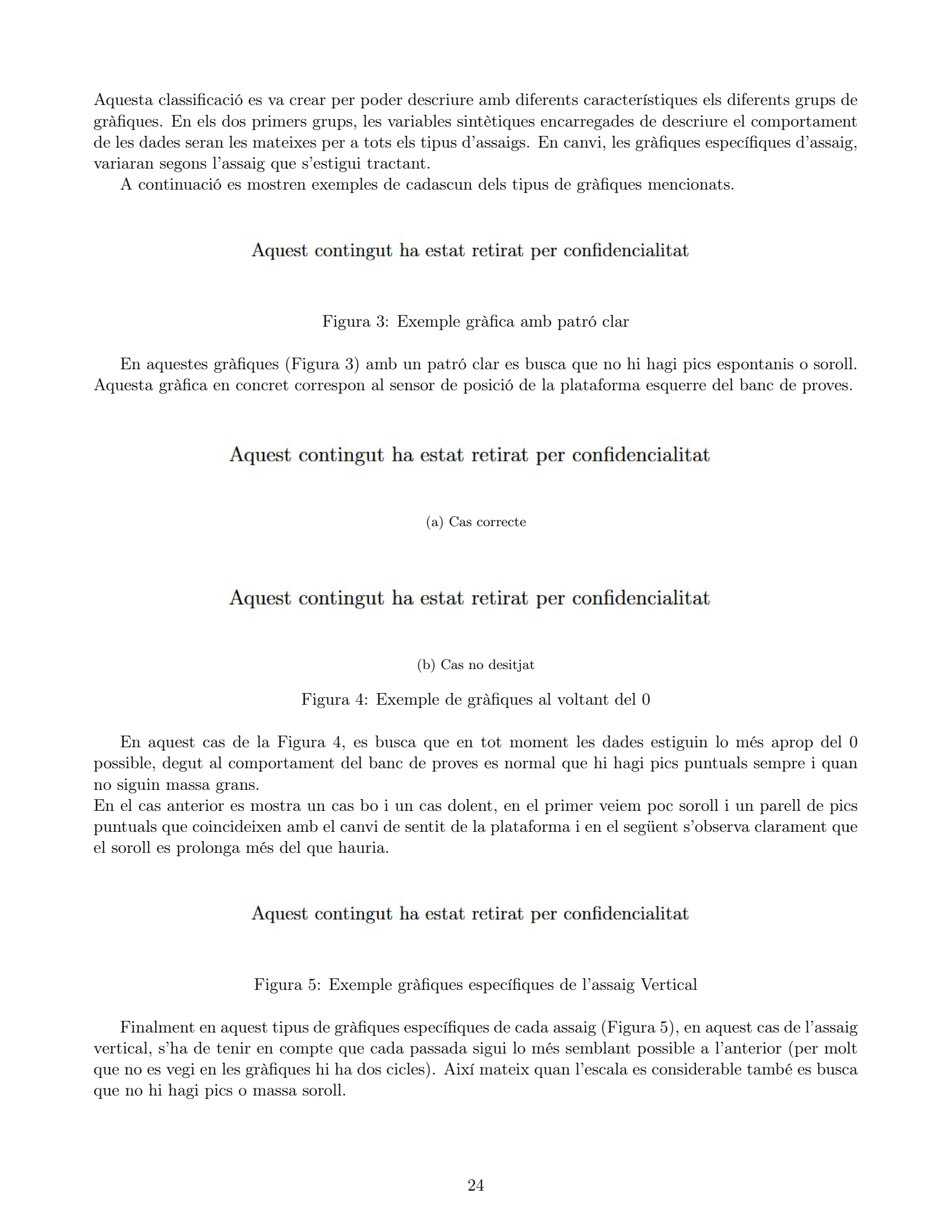

Figura 3: Exemple gràfica amb patró clar

En aquestes gràfiques (Figura 3) amb un patró clar es busca que no hi hagi pics espontanis o soroll. Aquesta gràfica en concret correspon al sensor de posició de la plataforma esquerre del banc de proves.

(a) Cas correcte

(b) Cas no desitjat

Figura 4: Exemple de gràfiques al voltant del 0

En aquest cas de la Figura 4, es busca que en tot moment les dades estiguin lo més aprop del 0 possible, degut al comportament del banc de proves es normal que hi hagi pics puntuals sempre i quan no siguin massa grans.

En el cas anterior es mostra un cas bo i un cas dolent, en el primer veiem poc soroll i un parell de pics puntuals que coincideixen amb el canvi de sentit de la plataforma i en el següent s'observa clarament que el soroll es prolonga més del que hauria.

Figura 5: Exemple gràfiques específiques de l'assaig Vertical

Finalment en aquest tipus de gràfiques específiques de cada assaig (Figura 5), en aquest cas de l'assaig vertical, s'ha de tenir en compte que cada passada sigui lo més semblant possible a l'anterior (per molt que no es vegi en les gràfiques hi ha dos cicles). Així mateix quan l'escala es considerable també es busca que no hi hagi pics o massa soroll.

3.2.3 Variables Sintètiques

Cal esmentar que tot l'estudi s'ha realitzat només amb l'assaig *Vertical*, ja que si s'obtenia un bon resultat per aquest primer cas el projecte tenia continuació; però, d'altra banda, si la precisió del model no era l'esperada per aquest primer cas ja no caldria investigar els altres tipus d'assaigs.

Per fer una primera avaluació del model s'ha decidit associar als diferents grups de gràfiques diferents característiques senzilles però suficientment descriptius per comprovar si el model te capacitat de millora o no.
Pel primer grup de gràfiques sinusoidals la variable escollida ha estat la següent:
Sigui $D = \{x_1, x_2, \ldots, x_n\}$ un vector de n dades, corresponent a qualsevol dels canals (gràfiques) a analitzar.

- Detecció de pics **(pd)**: Aquesta variable ajuda a detectar pics sobtats en un senyal que hauria de ser suau i de caràcter periòdic. Es defineix la variable com:

$$pd = \frac{max(\Delta D)}{\sigma(\Delta D)}$$

  On $\Delta D = \{\frac{x_2 - x_1}{t_2 - t_1}, \ldots, \frac{x_n - x_{n-1}}{t_n - t_{n-1}}\}$ és la derivada numèrica del vector de dades respecte el temps i σ representa la funció desviació estàndard. La construcció d'aquesta variable permet detectar si hi ha pics sobtats en les dades o si hi ha massa dispersió.

Per a les gràfiques que estan al voltant de 0 s'han escollit:

- Suma de les dades al quadrat **(area)**: Aquesta variable permet quantificar la presència de soroll i identificar si aquest soroll és puntual o generalitzat. Definida com:

$$area = \sum_{i=0}^{n} x_i^2$$

  Com es pot observar en aquesta variable, si un punt qualsevol del conjunt de dades, x_i, es troba al voltant del 0 la seva contribució serà molt petita, en canvi a mesura que se separi més, més gran serà la contribució. Per tant, a més *area* més punts se separaran del 0, o més lluny estaran del 0, cosa que indicaria pics i/o soroll.

- Valor mínim **(Min)** i el valor màxim **(Max)** del conjunt de dades. Definides com:

$$min(D) = \min_{1 \leq i \leq n} x_i$$

$$max(D) = \max_{1 \leq i \leq n} x_i$$

  Aquestes dues variables complementen l'anterior, descrivint així si el màxim o el mínim tenen valors absoluts molt grans i per tant la variable de l'*area* també te un valor alt, o, en cas contrari, si el valor de l'*area* es gran degut a una contribució més o menys uniforme de tots els punts i no de contribucions grans puntuals.

Finalment en el tercer grup de gràfiques específiques, s'ha de desglossar el contingut en els diferents assaigs, com només s'ha estudiat el cas de l'assaig *Vertical* només es mencionarà aquest.

1. **Vertical Motion**:
   - **Detecció de pics (pd)**

- **Correlació de cicles (*cycle correlation*)**: S'avalua la similitud entre dues meitats consecutives de la senyal per comprovar la consistència del cicle d'assaig. Es defineixen les dues meitats de la senyal:

  $$D_1 = \{x_1, \ldots, x_m\}, \quad D_2 = \{x_{m+1}, \ldots, x_{2m}\}$$

  on $m = \lfloor n/2 \rfloor$.

  $$\text{corr}_{\text{cicles}} = \text{corr}(D_1, D_2) = \text{corr}(\{x_1, \ldots, x_{\lfloor n/2 \rfloor}\}, \{x_{\lfloor n/2 \rfloor + 1}, \ldots, x_n\})$$

  Es calcula la correlació de Pearson:

  $$\rho = \frac{\text{Cov}(D_1, D_2)}{\sigma_{D_1} \cdot \sigma_{D_2}} = \text{corr}(D_1, D_2)$$

  Aquesta mesura quantifica la semblança entre dues meitats consecutives de la senyal. Una alta correlació indica una resposta periòdica i estable entre cicles d'anada i tornada, característica de senyals de qualitat.

  Una correlació elevada indica que el comportament de la senyal es repeteix d'un cicle a l'altre, característica pròpia d'un senyal correcte.

- **Diferència d'anada i tornada (*cycle sum difference*)**: Es calcula la diferència entre la suma de valors en la primera quarta part i la segona quarta part de la senyal.

  $$\Delta_{\text{cicle}} = \sum_{i=1}^{n/4} x_i - \sum_{i=n/4+1}^{n/2} x_i$$

  Aquesta diferència pot indicar asimetries entre les fases d'anada i tornada del moviment vertical.

- **Mitjana de la derivada**: Es calcula la mitjana dels valors absoluts de la derivada per quantificar el canvi mitjà del senyal.

  $$\mu_{|f'|} = \frac{1}{n-1} \sum_{i=1}^{n-1} |x_{i+1} - x_i|$$

  Aquesta variable dona una idea de la velocitat mitjana de canvi del senyal, útil per identificar assajos molt suaus o molt sobtats.

- **Desviació estàndard de la derivada**:

  $$\sigma_{|f'|} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n-1} \left(|x_{i+1} - x_i| - \mu_{|f'|} \right)^2}$$

  Mesura la variabilitat en la velocitat de canvi del senyal. Una desviació alta pot indicar soroll o inconsistències en la mesura.

Donades totes aquestes noves variables, s'ha de tenir en compte que aquesta ha estat la selecció de característiques inicial, donades les condicions limitades del conjunt de dades, es van prioritzar mètriques simples però informatives, més endavant s'estudiaran noves possibilitats per ajudar al rendiment del model.

4 Comparació d'Enfocaments: No Supervisat vs Supervisat

Per a l'anàlisi del problema s'ha decidit comparar dos punts de vista diferents: entrenar un model no supervisat (sense dades etiquetades), aprofitant que el nombre de dades totals és superior al nombre de dades etiquetades disponibles, o bé optar per entrenar un model supervisat, ajudant al model a realitzar una classificació més precisa i alineada amb l'experiència humana.

- **Anàlisi de Gràfiques Individuals (No Supervisat):** Aquest primer enfocament analitza cada gràfica de l'assaig per separat. Cada gràfica es descriu mitjançant un conjunt de característiques i es construeix un sistema format per diversos models petits, cadascun entrenat sobre una gràfica concreta. L'objectiu és detectar comportaments anòmals de manera individual per cada gràfica.

  Aquest tipus d'anàlisi requereix un model no supervisat, ja que les etiquetes disponibles són a nivell d'assaig, no de gràfica. Això pot provocar que, en casos on només una gràfica ha fallat, totes les gràfiques d'aquell assaig siguin etiquetades com a dolentes, introduint soroll en l'entrenament si es tractés com a problema supervisat. Per tant, aquest plantejament s'ajusta millor a models no supervisats, com l'algoritme **Isolation Forest**, especialitzat en detecció d'anomalies.

  Un avantatge d'aquest enfocament és que permet entrenar amb un volum de dades molt més gran, ja que inclou assaigs no etiquetats. No obstant això, el seu punt feble és que la classificació d'anomalies pot ser poc precisa, atès que la definició d'"assaig dolent" depèn de múltiples factors contextuals (tipus de vehicle, errors del banc de proves, temps disponible per projecte, etc.), fent que la classificació sigui força subjectiva.

- **Classificació Global d'Assaigs (Supervisat):** Aquest segon plantejament tracta l'assaig com una unitat completa, combinant totes les seves gràfiques per formar un vector global de característiques. Per exemple, si un assaig té 20 gràfiques i cada gràfica aporta 5 característiques, l'assaig estarà descrit amb 100.

  En aquest cas, es disposa de menys dades per entrenar, ja que només s'utilitzen assaigs amb etiquetes de classificació fiables. Tot i així, el model pot aprofitar aquestes etiquetes per aprendre patrons més realistes, alineats amb la classificació humana. Aquest enfocament utilitza un algoritme de classificació supervisat, el **Random Forest Classifier**, que és robust davant dades sorolloses i proporciona interpretabilitat mitjançant la importància de les variables.

4.1 Justificació de la comparació

Inicialment, pot no quedar clar si es disposa de prou dades etiquetades i de si la classificació aporta valor a l'entrenament. *Isolation Forest* permet treballar en aquest escenari i identificar casos anòmals sense necessitat d'etiquetes.
D'altra banda es desconeix si els casos classificats com a dolents son fàcils d'aïllar respecte els casos bons, o per contra si la línia divisòria entre els dos casos és fina. En el cas de que els assaigs no vàlids siguin semblants als vàlids, o no hi hagi un llindar clar per fer la classificació, l'etiquetatge dels assaigs ajudarà al model supervisat *Random Forest* fer una classificació més semblant a la humana.
En resum:

- **Isolation Forest** parteix de la hipòtesi que les anomalies són més fàcils d'aïllar i segmenta els valors que difereixen de la norma.

- **Random Forest** fa ús d'etiquetes reals per construir múltiples arbres de decisió i combinar-ne la sortida via votació, aconseguint robustesa i una bona capacitat de generalització.

Aquesta comparació permet establir si el projecte ha d'avançar cap a models supervisats o si seria millor continuar amb tècniques no supervisades.

Models com SVM (*Support Vector Machine*), KNN (*K Nearest Neighbours*) o xarxes neuronals no són rellevants abans de decidir si el problema s'enfoca millor des del vessant supervisat o no supervisat. Per tant, la comparació inicial es limita a aquests dos algoritmes per prendre una decisió estructural.

A continuació es comentarà com s'han estructurat les dades en els dos plantejaments.

4.1.1 *Isolation Forest*

L'algorisme `Isolation Forest` [15] és un mètode no supervisat de detecció d'anomalies, especialment dissenyat per identificar instàncies poc freqüents o atípiques, el que es coneix com *outliers*, dins un conjunt de dades. A diferència d'altres mètodes basats en distàncies o densitats, la seva estratègia es fonamenta en la idea que les anomalies són més fàcils d'aïllar mitjançant particions recursives de l'espai de característiques.

El funcionament del model es basa en la construcció d'un bosc d'arbres binaris (*isolation trees*), creats de manera totalment aleatòria. En cada node intern d'un arbre, es selecciona una característica (*feature*) f de manera aleatòria i es tria un llindar de separació t entre els valors mínim i màxim observats de f. Aquesta partició divideix les dades en dos subconjunts: aquells amb $f < t$ i els amb $f \geq t$. Aquest procés es repeteix recursivament fins que cada instància queda aïllada o s'assoleix una profunditat màxima predefinida.

Les instàncies considerades anòmales solen trobar-se en regions escassament poblades de l'espai de característiques. A causa d'aquesta dispersió, són aïllades més aviat, és a dir, la seva profunditat mitjana $\mathbb{E}(h(x))$ dins els arbres del bosc és menor que la de les instàncies normals. Aquesta observació fonamenta la definició de l'índex d'anomalia (*anomaly score*) per a cada instància x:

$$s(x, n) = 2^{-\frac{\mathbb{E}(h(x))}{c(n)}}$$

on n és el nombre d'instàncies de l'arbre i $c(n)$ és una constant de normalització que representa la profunditat mitjana esperada d'un arbre binari amb n fulles, donada per:

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n} \quad \text{amb} \quad H(i) \approx \ln(i) + \gamma,$$

on γ és la constant d'Euler-Mascheroni (≈ 0.577). Això permet interpretar el valor de $s(x)$ com la probabilitat que x sigui una anomalia: valors propers a 1 indiquen alta probabilitat d'anomalia, mentre que valors propers a 0.5 suggereixen comportament normal.

Una de les grans virtuts d'`Isolation Forest` és la seva eficiència computacional. Tant la construcció dels arbres com el càlcul dels scores tenen una complexitat de $\mathcal{O}(n \log n)$, fet que el fa escalable a conjunts de dades molt grans. A més, no assumeix cap distribució prèvia de les dades, fet que el fa molt flexible i adaptable a situacions on les anormalitats no tenen una estructura estadística clara. També funciona bé amb dades d'alta dimensionalitat.

Tanmateix, aquest model també presenta certes limitacions. L'aleatorietat intrínseca pot afectar la consistència dels resultats si no es construeixen suficients arbres. Així mateix, la seva capacitat discriminativa pot reduir-se en entorns amb soroll alt o quan les anomalies són molt similars a les instàncies normals, és a dir, no són fàcilment separables mitjançant particions simples. També és sensible a la presència de moltes característiques poc informatives, fet que pot requerir una selecció o reducció prèvia de variables.

En el context d'aquest treball, on es busca identificar mostres anòmales sense disposar de supervisió explícita, `Isolation Forest` és especialment adequat. El problema presenta una distribució fortament desequilibrada, on les classes minoritàries (casos "dolents") poden ser assimilables a anomalies estadístiques. Per tant, l'aplicació d'aquest mètode permet detectar patrons inusuals sense necessitat d'etiquetes, ajudant a complementar l'anàlisi supervisada i millorant la robustesa general del sistema predictiu.

4.1.2 *Random Forest*

El model `Random Forest` és un algorisme supervisat basat en arbres de decisió. La seva lògica es fonamenta en la combinació de múltiples classificadors febles (arbres de decisió individuals) per construir un model fort, robust i menys susceptible al sobreajustament.

En concret, una `Random Forest` construeix un gran nombre d'arbres de decisió de forma independent utilitzant dues fonts principals d'aleatorietat: primer, selecciona aleatòriament, amb reemplaçament, subconjunts del conjunt d'entrenament (això es coneix com *bagging*, o *bootstrap aggregation*); segon, a cada node de l'arbre, només es considera un subconjunt aleatori de característiques per determinar la millor divisió. Aquesta estratègia introdueix diversitat entre els arbres i redueix la correlació entre ells, fet que millora la generalització del model.

Cada arbre de la bosc vota individualment per una classe, i la predicció final es determina per vot majoritari (en classificació) o mitjana (en regressió). La naturalesa col·lectiva d'aquest model li permet capturar millor patrons complexos i no lineals en les dades, alhora que minimitza la variància que afecta els arbres de decisió aïllats.

La probabilitat de classificació per una classe k és:

$$P(k|x) = \frac{1}{T} \sum_{t=1}^{T} I(h_t(x) = k)$$

on:

- T és el nombre total d'arbres.
- $h_t(x)$ és la predicció del t-è arbre.
- I és la funció indicador que val 1 si $h_t(x) = k$, i 0 si no.

Un avantatge important de `Random Forest` és la seva robustesa davant dades sorolloses i la seva capacitat per tractar conjunts de dades amb moltes característiques, incloent-hi dades categòriques i contínues. A més, ofereix informació valuosa sobre la importància relativa de les característiques (*feature importance*), basada habitualment en la disminució de la impuresa (com la mètrica Gini) al llarg dels arbres.

Tanmateix, el model no és exempt de limitacions. Tot i que generalment és menys propens al sobreajustament que un arbre únic, pot arribar a sobreajustar-se si es construeixen arbres molt profunds amb un nombre insuficient d'instàncies. A més, pot resultar menys interpretable que un arbre simple, i la seva eficiència computacional pot veure's compromesa quan s'incrementa molt el nombre d'arbres o de característiques.

En el context del problema abordat en aquest treball, on es pretén classificar assaigs entre casos "bons" i "dolents", `Random Forest` resulta especialment prometedor. El seu funcionament intern, basat en múltiples decisions no lineals, li permet capturar interaccions complexes entre variables que poden ser rellevants per a la detecció de patrons associats al mal funcionament. A més, el model tendeix a funcionar bé amb conjunts de dades desequilibrats si s'ajusta adequadament (per exemple, mitjançant pesos de classe o *sampling*). Això s'ha corroborat experimentalment, ja que ha mostrat un bon rendiment tant en el conjunt d'entrenament com de test, tot i que, com es veurà més endavant, pot presentar lleuger sobreajustament si no es controla la profunditat dels arbres o el nombre de variables seleccionades a cada divisió.

4.2 Estructuració de les Dades

Model No Supervisat - Isolation Forest (Gràfiques Individuals)

Com bé s'ha comentat, en aquest primer plantejament s'entrena un petit model per a cadascuna de les diferents gràfiques amb les seves característiques.

- Entrenament a nivell de gràfica, considerant cada una com una observació independent.
- No es requereixen etiquetes, ja que si es vol plantejar el problema a nivell gràfica les etiquetes al ser de nivell assaig només aportarien soroll a l'entrenament, ja que un assaig pot tenir totes les gràfiques bé excepte una i per aquest fet ja es considerarien totes les gràfiques com a errònies.
- Entrenament amb un major nombre de dades respecte al model supervisat.

A continuació es mostra una il·lustració de com s'han estructurat les dades per a l'entrenament del model no supervisat.

ensayo_id	graph	graph_id	pd	class
1_202502:	LfVertPosi	0	1,063661	0
1_202502:	LfVertPosi	0	1,063661	0
20240905_	LfVertPosi	0	1,06558	0
20240905_	LfVertPosi	0	1,06558	0
20240910_	LfVertPosi	0	1,066677	0
20240910_	LfVertPosi	0	1,066677	0
20240910_	LfVertPosi	0	1,086724	0
20240910_	LfVertPosi	0	1,086724	0
2_202502:	LfVertPosi	0	1,156603	0
2_202502:	LfVertPosi	0	1,156603	0

graph_0 | graph_1 | graph_2 | graph_3 | graph_4 | graph_5 | graph_6 | graph_7 | graph_8 | grap ...

Figura 6: Estructura de l'arxiu de dades sintètiques de l'assaig *Vertical* pel model *Isolation Forest*

Com bé s'observa, el document està format per tantes pàgines com diferents tipus de canals de dades (gràfiques) a analizar conté l'assaig; Cada pàgina del document correspon a una gràfica diferent i en cada pàgina hi apareix la següent informació:

- *ensayo_id*: Nom de l'assaig.
- *graph*: nom de la gràfica (en una mateixa pàgina el nom serà el mateix sempre).
- *graph_id*: Índex associat a cada gràfica per ordre d'aparició.
- *Columnes de característiques*: en aquest primer cas només trobem la característica de detecció de pics *pd*, ja que aquesta gràfica correspon a les de comportament sinusoidal i per tant només està descrita per aquesta variable.
- *Class*: Finalment aquesta columna indica si la gràfica pertany a un assaig que ha estat classificat com a bo (0) o dolent (1); En el cas de que l'assaig no tingui cap classificació, el contingut de la columna *Class* serà irrellevant ja que tots els arxius no classificats estan identificats. Aquesta columna servirà posteriorment per poder fer una anàlisis dels resultats predits del model no supervisat, només tenint en compte les dades etiquetades.

Model Supervisat - Random Forest (Assaigs Complets)

- Cada assaig es representat per les característiques de les seves dades (gràfiques).
- Només es consideren assaigs amb etiquetes fiables.
- Les dades etiquetades es divideixen en un **80% per entrenament** i **20% per test**.

Seguidament es mostra una imatge de l'estructura que prenen les dades per a l'entrenament d'aquest model supervisat.

	ensayo_id	pd_1	pd_2	pd_3	pd_4	pd_5	pd_6	pd_7	pd_8	pd_9	pd_10	max_11	min_11	area_11	max_12
0	1_202502	1,06366	1,06499	2,14659	2,18794	1,48773	1,43702	3,95827	3,61071	3,37065	3,56398	0,00019	-0,00022	3,3E-05	0,00017
1	1_202502	1,06366	1,06499	2,14659	2,18794	1,48773	1,43702	3,95827	3,61071	3,37065	3,56398	0,00019	-0,00022	3,3E-05	0,00017
2	20240905	1,06558	1,06304	2,1269	2,11742	1,4225	1,31628	4,13219	2,52635	2,78767	2,99244	0,0003	-0,00026	5,4E-05	0,00024
3	20240905	1,06558	1,06304	2,1269	2,11742	1,4225	1,31628	4,13219	2,52635	2,78767	2,99244	0,0003	-0,00026	5,4E-05	0,00024
4	20240910	1,06668	1,06304	2,11642	2,09145	1,51105	1,3042	3,66608	2,27396	3,01809	3,15139	0,00023	-0,0003	5,5E-05	0,00017
5	20240910	1,06668	1,06304	2,11642	2,09145	1,51105	1,3042	3,66608	2,27396	3,01809	3,15139	0,00023	-0,0003	5,5E-05	0,00017
6	20240910	1,08672	1,08242	2,1791	2,16654	1,46911	1,32149	3,63488	2,46739	2,8293	3,02589	0,00021	-0,00022	3,7E-05	0,0002
7	20240910	1,08672	1,08242	2,1791	2,16654	1,46911	1,32149	3,63488	2,46739	2,8293	3,02589	0,00021	-0,00022	3,7E-05	0,0002
8	2_202502	1,1566	1,15232	2,01787	2,05645	1,7217	1,55779	4,47357	4,66413	1,50987	1,53157	0,00031	-6,4E-05	6,1E-05	-8,1E-05
9	2_202502	1,1566	1,15232	2,01787	2,05645	1,7217	1,55779	4,47357	4,66413	1,50987	1,53157	0,00031	-6,4E-05	6,1E-05	-8,1E-05
10	6_202502	1,06341	1,06552	2,16016	2,18793	1,49715	1,51727	4,38153	3,29739	3,46612	3,62771	0,0002	-0,00027	5,2E-05	0,00015
11	6_202502	1,06341	1,06552	2,16016	2,18793	1,49715	1,51727	4,38153	3,29739	3,46612	3,62771	0,0002	-0,00027	5,2E-05	0,00015
12	8_202502	1,15458	1,15543	2,15272	2,17839	1,49806	1,29071	3,29281	3,59914	1,84342	1,87442	0,00041	-0,00029	0,00011	1,9E-05
13	8_202502	1,15458	1,15543	2,15272	2,17839	1,49806	1,29071	3,29281	3,59914	1,84342	1,87442	0,00041	-0,00029	0,00011	1,9E-05
14	AM_003_F	1,08562	1,08864	3,26414	3,54745	2,04015	2,3187	5,08018	3,98632	2,86182	2,83676	0,00219	-0,00176	0,00049	0,00475
15	AM_003_F	1,08397	1,09031	3,38765	4,6561	2,47216	2,74142	6,4837	7,39443	2,92828	2,84661	0,00182	-0,00192	0,00053	0,00277
16	AM_003_F	1,24941	1,25096	3,78602	4,53598	1,9838	2,31812	4,24376	5,0209	1,92486	2,097	0,00216	-0,00188	0,00048	0,00499
17	AM_003_F	1,38045	1,3692	4,3993	6,09751	2,20728	2,42812	5,65243	3,86015	1,63553	1,82269	0,00181	-0,00192	0,00063	0,00259

Figura 7: Estructura de l'arxiu de dades sintètiques de l'assaig *Vertical* pel model *Random Forest*

Com bé es pot observar, en aquesta nova distribució que prenen les dades, ja no apareixen les gràfiques, ja que les variables sintètiques que abans descrivien les gràfiques individuals ara queden tots agrupats i passen a descriure l'assaig. En aquest document només es presenta una pàgina que conté tots els assaigs i d'aquests es pot treure la següent informació:

- *ensayo_id*: Nom de l'assaig
- *Columnes de característiques*: Totes les variables sintètiques encarregades de descriure l'assaig. La il·lustració s'atura en el *feature* anomenat *max* de la gràfica número 12; però, va més enllà fins arribar a la columna de *class*. Cada *feature* té indicat a quin índex de gràfica pertany.
- *Class*: Etiqueta de classificació de l'assaig. Bo (0) i dolent (1).

L'algorisme d'*Isolation Forest* s'ha entrenat amb un total de 2750 arxius de dades dels quals 1271 tenen classificació. El 80% d'aquest últim conjunt ha estat l'encarregat d'entrenar el model supervisat de *Random Forest*, l'altre 20% forma part del conjunt de validació.

Els codis de la consturcció de les variables sintètiques i estructuració del document de dades per a l'entrenamnet de ambdós models es trobaran a la secció 9.6 de l'annex.

4.3 Entrenament i Validació

Un cop estructurades les dades en els dos casos es van entrenar els dos models.

Tant per a la detecció d'anomalies amb *Isolation Forest* com per a la classificació binària amb *Random Forest*, s'han dut a terme una sèrie de passos comuns relacionats amb la preparació de les dades i la implementació del model. Aquests punts compartits són essencials per garantir una base consistent i robusta per als entrenaments.

Llibreries Utilitzades

Per a l'entrenament de tots dos models s'han utilitzat les següents llibreries de Python:

- **pandas**: Manipulació de dades tabulars, lectura d'arxius Excel amb múltiples fulles i gestió de columnes.
- **numpy**: Operacions numèriques eficients amb arrays i matrius.
- **matplotlib.pyplot**: Visualització de gràfiques per interpretar els resultats.
- **seaborn**: Per generar gràfics com la matriu de confusió i les importàncies de les variables.
- **sklearn.metrics**: Per calcular mètriques de rendiment com l'exactitud, precisió, sensibilitad, *F1-score*, i la corba ROC.

Aquestes dues últimes llibreries de normal no serien necessàries en un model no supervisat com *Isolation Forest*, perquè, al no tenir un conjunt d'entrenament específic diferent al conjunt de validació, aquestes mètriques no es podrien calcular, però en aquest cas, cal recordar que el model no supervisat ha estat parcialment entrenat amb dades que si tenen classificació i per tant les mètriques es poden calcular sobre aquest conjunt concret.

Preparació de les Dades

En ambdós casos:

- Les dades s'han extret d'un arxiu Excel, el format de cadascun d'aquests arxius s'ha esmentat en la secció 4.2 Estructuració de Dades.
- Les dades s'han organitzat en `DataFrames` per facilitar la seva manipulació.
- S'han descartat les columnes no rellevants per a l'entrenament (com `ensayo_id`, `graph`, etc.), mantenint només les variables sintètiques significatives. En el cas d'*Isolation Forest* la columna corresponent a l'etiqueta (*Class*) de cada gràfica no serveix per a l'entrenament, només per a l'avaluació posterior.

Cal esmentar que en aquesta primera fase del projecte no es va fer servir el mètode de *Cross Validation* per tal d'aconseguir els millors paràmetres del model de *Random Forest*, ja que *Isolation Forest* al ser un model no supervisat, no disposa d'un conjunt d'entrenament i un conjunt de validació per separat sinó que totes les dades són a la vegada d'entrenament i de validació, per tant aquesta metodologia no li és aplicable. Per fer la comparació més justa es va decidir 'limitar' el model de *Random Forest* en aquest sentit.

4.3.1 Entrenament amb *Isolation Forest*

Per dur a terme la detecció d'anomalies en les diferents gràfiques experimentals, s'ha optat per utilitzar el model `Isolation Forest` de la llibreria `sklearn.ensemble`.

Hiperparàmetres

Les dades s'han carregat a partir d'un fitxer Excel amb múltiples fulles (una per cada tipus de gràfica), tal com es mostra a la figura 6. Cada fulla s'ha llegit com un *DataFrame* i se n'han seleccionat només les variables sintètiques (excloent-ne identificadors com `ensayo_id`, `graph` i `Class`, ja que aquesta última columna només servirà per fer l'avaluació posterior del model).

Per a cada conjunt de dades, s'ha entrenat un model d'`IsolationForest` amb els següents hiperparàmetres:

- **contamination = 0.05**: Indica que s'espera que un 5% de les mostres siguin anòmales. Aquest valor és adequat en el nostre cas, ja que la proporció de dades sospitoses és baixa però existent. Aquest paràmetre ajusta automàticament el llindar de decisió que separa observacions normals d'anòmales.
- **random_state = 42**: Estableix una llavor aleatòria per garantir la reproductibilitat dels resultats. Això assegura que, en executar el model diverses vegades amb el mateix conjunt de dades, s'obtinguin els mateixos resultats.
- **n_estimators = 100**: Nombre d'arbres del bosc. El valor per defecte és suficient en aquest cas, atès que els conjunts de dades no són excessivament grans i no es requereix una complexitat elevada per detectar patrons d'anomalia.
- **max_samples = 'auto'**: Especifica el nombre màxim de mostres utilitzades per construir cada arbre. El valor `'auto'` indica que s'utilitza el mínim entre 256 i el nombre total de mostres, permetent així una execució eficient en conjunts grans sense comprometre l'eficàcia del model.
- **max_features = 1.0**: Totes les variables sintètiques es tenen en compte en la construcció de cada arbre. Això és especialment rellevant en aquest cas, ja que cada variable aporta informació potencialment útil per detectar anomalies.

Avaluació del model

Tot i que el model `Isolation Forest` és no supervisat, en aquest cas ha estat parcialment entrenat amb dades etiquetades, cosa que permet avaluar el seu rendiment com si es tractés d'un model supervisat només per aquest conjunt.
Per això, s'ha comparat la classificació generada pel model amb les etiquetes reals dels assaigs. Per dur-ho a terme, s'ha assumit que si alguna de les gràfiques d'un assaig ha estat marcada com a anòmala pel model, tot l'assaig es considera defectuós, mateix procediment que duen a terme els experts del departament. Aquesta regla de decisió permet obtenir una classificació binària a nivell d'assaig, que es pot comparar amb la classificació real.
Amb aquesta estratègia, s'ha pogut construir una matriu de confusió i calcular diverses mètriques de classificació com ara:

- **Exactitud (*Accuracy*)**: Proporció total de classificacions correctes.
- **Precisió**: Proporció de prediccions positives que realment ho són.
- **Sensibilitat (*Recall*)**: Proporció de casos positius correctament detectats.

- ***F1-score***: Mitjana harmònica entre la precisió i la sensibilitat, útil en casos de dades desbalancejades.

Aquestes mètriques permeten una comparació directa amb els resultats obtinguts amb el model de *Random Forest*, i aporten informació quantitativa sobre la capacitat de detecció d'anomalies del model `Isolation Forest` en un entorn semisupervisat.

4.3.2 Entrenament amb *Random Forest*

El model de *Random Forest* forma part dels algoritmes supervisats i és especialment útil per a tasques de classificació. Es basa en la construcció d'un conjunt d'arbres de decisió entrenats sobre subconjunts aleatoris del conjunt de dades. La seva robustesa i capacitat de generalització el fan ideal per problemes amb variables diverses i relacions no lineals.

Divisió del Conjunt de Dades

El conjunt de dades etiquetades s'ha dividit en dues parts:

- **Train (80%)**: Per entrenar el model.
- **Test (20%)**: Per avaluar la capacitat predictiva del model sobre dades no vistes durant l'entrenament.

Aquesta divisió garanteix una validació objectiva del rendiment del model i evita el risc de sobreajustament (overfitting).

Hiperparàmetres

- **n_estimators = 200**: Nombre d'arbres del bosc. S'ha escollit un valor relativament alt per millorar l'estabilitat del model. En problemes com el nostre, amb dades moderades, aquest nombre és suficient per assolir una bona generalització sense arribar al sobreajustament.
- **class_weight = 'balanced'**: Compensa automàticament el desequilibri entre classes, ajustant els pesos inversament a la seva freqüència. Aquesta estratègia és fonamental quan una classe (p. ex. "anomalia") és menys representada, per evitar que el model ignori els casos minoritaris.
- **random_state = 42**: Per garantir la reproductibilitat dels resultats entre execucions.

Avaluació del model

Un cop entrenat el model, s'han generat prediccions sobre el conjunt de test. La seva precisió s'ha avaluat mitjançant les següents mètriques:

- **Matriu de confusió**: Permet visualitzar la relació entre les prediccions correctes i les incorrectes per a cada classe.
- **Exactitud, precisió, sensibilitat, *F1-score***: Mètriques estàndard de classificació que permeten quantificar el rendiment global del model.

4.3.3 Resultats *Isolation Forest*

El model `Isolation Forest` s'ha entrenat amb un conjunt de 2750 assaigs, dels quals 1255 estan etiquetats. Tot i que és un model no supervisat, el fet de disposar de dades parcialment etiquetades ha permès fer una avaluació objectiva del seu rendiment.
Per fer aquesta avaluació, s'ha considerat que si alguna de les gràfiques d'un assaig és detectada com a anòmala pel model, tot l'assaig es classifica com a defectuós. A continuació, es descarten els assaigs sense etiqueta real per poder comparar les prediccions del model amb la classificació real i construir una matriu de confusió (Figura 8).

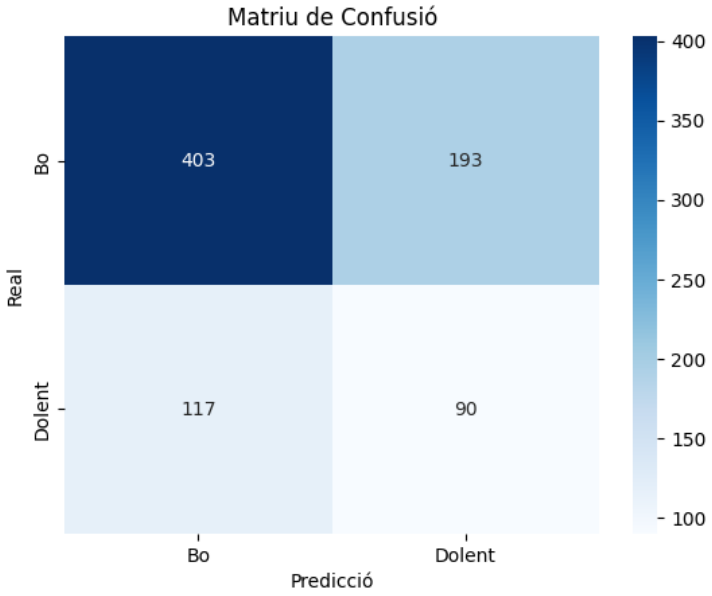

Figura 8: Matriu de confusió - *Isolation Forest*

Amb la matriu de confusió obtinguda es poden calcular les següents mètriques:

Mètrica	**Valor**
Accuracy	0.6139
Precision	0.3180
Recall	0.4348
F1-Score	0.3673

Taula 2: Resultats de l'avaluació del model amb Isolation Forest

Aquests resultats mostren que el model és capaç de detectar part dels assaigs defectuosos, tot i que presenta una precisió baixa (molts falsos positius) i un F1-score moderat. Malgrat tot, pot tenir utilitat com a eina preliminar de detecció d'anomalies en entorns on no es disposa d'etiquetes.

4.3.4 Resultats *Random Forest*

El model `Random Forest` s'ha entrenat sobre un conjunt completament etiquetat, utilitzant el 80% dels 1255 assaigs disponibles per entrenament i el 20% restant per a validació (test).

La predicció del model es va comparar amb les etiquetes reals, generant la següent matriu de confusió (Figura 9):

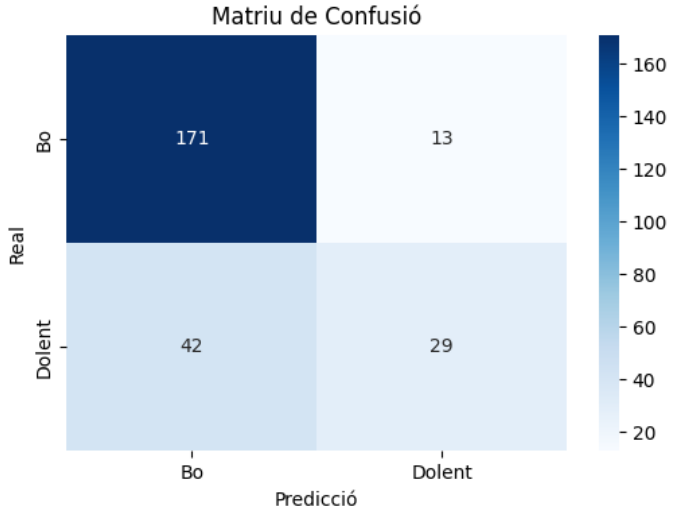

Figura 9: Matriu de confusió - *Random Forest*

A partir d'aquests resultats, es poden fer les següents observacions:

Mètrica	Valor
Accuracy	0.7843
Precision	0.6905
Recall	0.4085
F1-Score	0.5133

Taula 3: Resultats de classificació del model Random Forest

El model presenta una alta **precisió** en detectar assaigs defectuosos i una **exactitud general** significativa. No obstant això, la **sensibilitat** continua sent relativament baix, indicant que el model encara deixa escapar una part dels assaigs dolents. Malgrat això, el **F1-score** mostra un equilibri acceptable entre precisió i sensibilitat.

4.4 Comparació i Discussió de Resultats

Per tal de valorar quin model s'ajusta millor al problema de classificació d'assaigs com a bons o defectuosos, s'ha realitzat una comparació entre dos enfocaments diferents: **Isolation Forest** (no supervisat) i **Random Forest** (supervisat). A continuació es mostren les mètriques obtingudes en l'avaluació de cadascun:

Mètrica	**Isolation Forest**	**Random Forest**
Accuracy	0.6139	0.7843
Precision	0.3180	0.6905
Recall	0.4348	0.4085
F1-Score	0.3673	0.5133

Taula 4: Comparativa de mètriques entre Isolation Forest i Random Forest

Més enllà de les mètriques, és important tenir en compte el context en què s'han aplicat:

Taula 5: Comparació contextual dels models

Aspecte	**Isolation Forest (no supervisat)**	**Random Forest (supervisat)**
Tipus d'aprenentatge	No supervisat	Supervisat
Unitat d'anàlisi	Gràfica individual	Assaig complet (conjunt de gràfiques)
Necessitat d'etiquetes	No	Sí
Volum de dades d'entrenament	∼2750 gràfiques (totes disponibles)	∼1004 assaigs (80% de 1255 etiquetats)
Dades d'avaluació	Subset de 1255 assaigs etiquetats	20% del conjunt etiquetat (∼251 assaigs)

Les conclusions principals que se'n poden extreure són:

- El model **Random Forest** obté millors resultats en exactitud, precisió i *F1-score*, mostrant-se més fiable a l'hora de classificar correctament assaigs defectuosos quan es disposa d'etiquetes.
- Tot i això, l'**Isolation Forest** obté una sensibilitat lleugerament superior (0.4348 vs 0.4085), fet que indica una millor capacitat per detectar la totalitat de casos dolents (encara que augmenti el nombre de falsos positius). Aquesta propietat pot ser valuosa en entorns on la prioritat és no passar per alt cap defecte.
- L'**Isolation Forest**, en ser no supervisat, permet aprofitar tot el conjunt de dades disponibles, encara que no estiguin etiquetades. És ideal com a eina de detecció d'anomalies generals, especialment útil en fases preliminars o en temps real, on no es disposa d'etiquetatge o aquest pot ser poc fiable.
- El **Random Forest**, gràcies al seu entrenament supervisat, aprèn patrons específics a partir dels criteris humans i proporciona classificacions més precises quan les etiquetes són consistents i de qualitat.
- És rellevant considerar que les etiquetes poden contenir **soroll o subjectivitat**, especialment en entorns de proves de prototips on les condicions poden variar i on no sempre hi ha un criteri homogeni de classificació.

Tot i que el model d'`Isolation Forest` ha demostrat ser útil per detectar patrons anòmals de forma no supervisada, el model `Random Forest` té una capacitat de classificació superior quan es disposa d'un conjunt etiquetat de dades, fins i tot amb quasi un terç del volum de dades d'entrenament del model d'`Isolation Forest`, 1004 contra 2750. Per tant, en el context actual, on la línia divisòria entre un assaig vàlid i un altre no vàlid a vegades no es del tot clara, les dades etiquetades ajuden al model a fer aquesta distinció de forma més humana i no tant sistemàtica i per tant el model supervisat **Random Forest** és la millor opció per abordar una classificació realista dels assaigs.

Per altra banda, amb més temps d'investigació es podria plantejar la idea de desenvolupar una combinació dels dos models per tal d'aprofitar els punts forts de cadascun.

5 Estudi Aprofundit del Model Random Forest

Un cop escollit quin mètode d'entrenament que sembla ser més adient per al problema tractat, s'analitzarà de forma més detallada l'entrenament i els resultats del model *Random Forest*.
Per analitzar en profunditat el comportament del model, s'han considerat diferents mètriques i visualitzacions que permeten comprendre millor com classifica i fins a quin punt aprèn correctament de les dades.

5.1 Importància de les Característiques

Durant el desenvolupament del model `Random Forest`, s'han realitzat diversos entrenaments emprant conjunts de característiques diferents, amb l'objectiu de determinar si una selecció més àmplia, acotada o una transformació alternativa de les variables podria millorar el rendiment global del model. Tot i aquests intents, els resultats obtinguts en termes de `F1-score` i capacitat de generalització han estat sistemàticament inferiors respecte als que s'han aconseguit amb la configuració mencionada a l'apartat 3.2.3 *Variables sintètiques*.
Per aquest motiu, tots els entrenaments definitius, tant els entrenaments vistos prèviament del `Isolation Forest` i del `Random Forest` com els següents entrenaments que es comentaran de models supervisats alternatius, han estat realitzats amb aquest conjunt de variables sintètiques.

A continuació es mostra la importància relativa de les característiques definitives. Aquesta informació permet identificar quins paràmetres o mesures tenen més pes en la decisió del model.
Com s'ha mencionat en la secció 4.2 Estructuració de dades, concretament a l'apartat de "Model Supervisat - Random Forest (Assaigs Complets)", els números que apareixen juntament amb el nom de les variables indiquen a quina gràfica pertanyen.

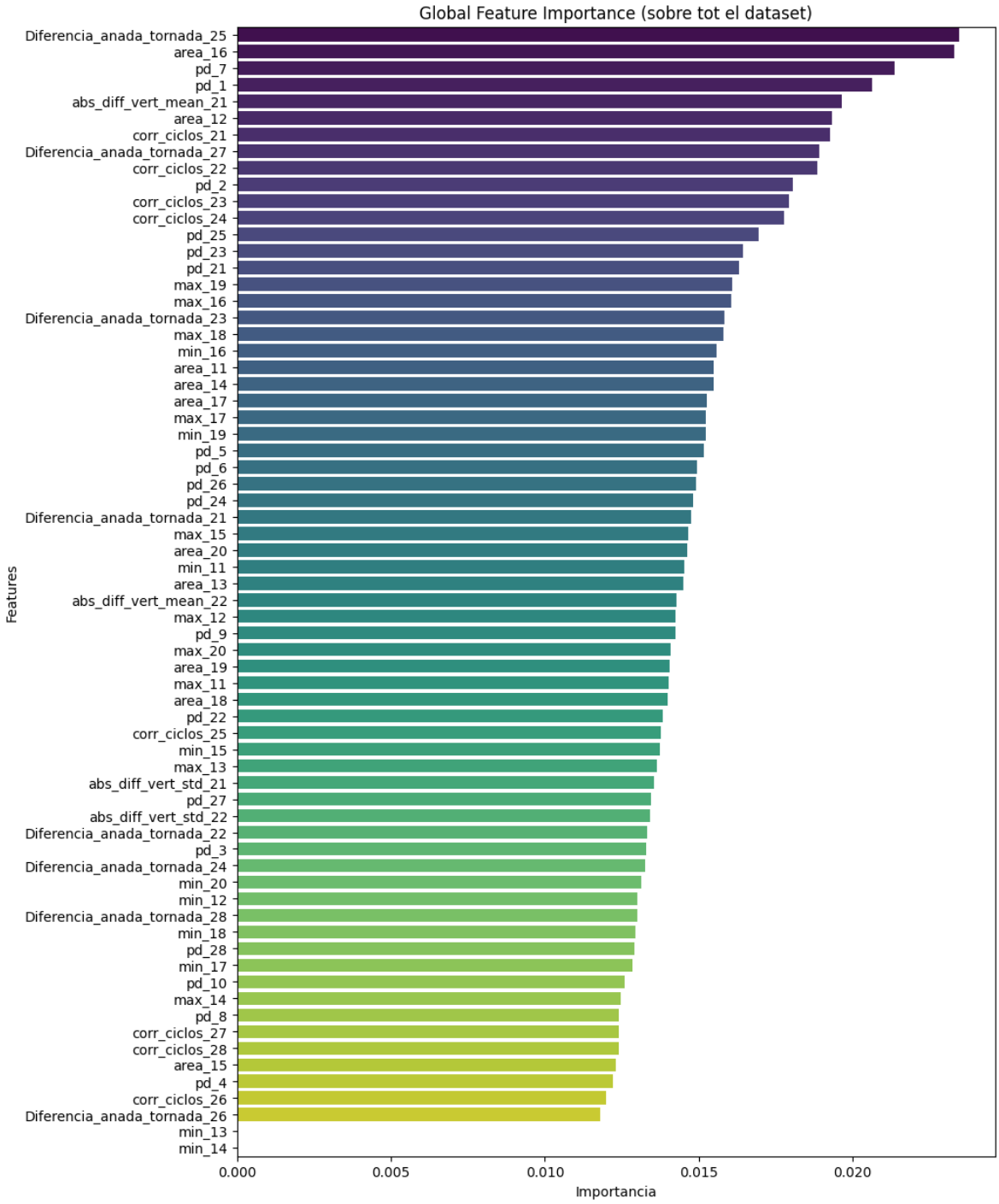

Figura 10: Importància de les característiques en el model Random Forest

Com s'observa en la gràfica mostrada (10), no hi ha una concentració extrema d'importància en poques variables, sinó que aquesta es reparteix de forma equilibrada entre un nombre elevat de característiques. Aquest comportament indica que no hi ha variables extremadament dominants, sinó que la decisió del model es basa en la contribució global de la majoria de les característiques.
Cal destacar que hi ha dos variables les quals no contribueixen en la decisió del model, això és degut a que tot i pertànyer a gràfiques del conjunt de dades que han d'estar al voltant del 0, abans de tractar amb les dades d'aquests canals, 13 i 14, el rang de valors es desplaça per a que el mínim comenci al 0, per tant *min_13* i *min_14* sempre seran igual a 0.

Algunes de les modificacions en les variables sintètiques han estat les següents:

Segon conjunt de variables

En aquest cas la única modificació respecte la configuració principal es que s'han suprimit les variables *min*, *max* de totes les gràfiques que pertanyen al grup de dades que han d'estar al voltant del 0 (aquest conjunt de gràfiques és el més problemàtic i per això s'ha estudiat més). Aquesta modificació va servir per comprovar si la variable *area* era suficient per captar el soroll de les gràfiques. A continuació es mostra la gràfica d'importància de les característiques.

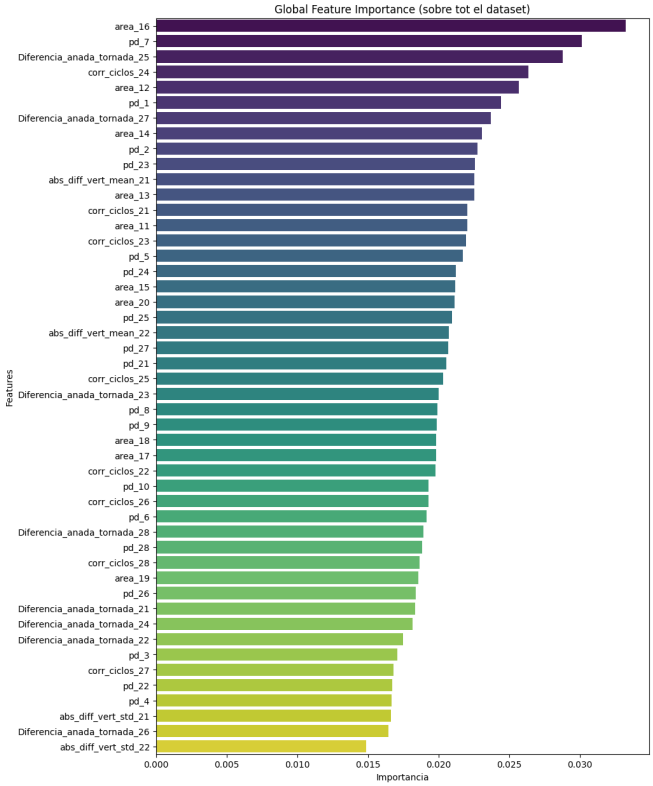

Figura 11: Importància del segon conjunt de característiques en el model Random Forest

Es pot observar que el patró en el gràfic es similar al del conjunt principal, vist anteriorment, on totes les variables sintètiques es reparteixen la capacitat de decisió del model.
Fins i tot les mètriques d'avaluació del model amb aquesta selecció de variables fou quasi igual de satisfactòria que amb el primer conjunt. A continuació es mostra una taula comparativa dels resultats:

Mètrica	**1er Conjunt**	**2n Conjunt**
Accuracy	0.7843	0.7804
Precision	0.6905	0.6744
Recall	0.4085	0.4085
F1-Score	0.5133	0.5088

Taula 6: Comparativa de mètriques entre conjunt de variables principal i segon conjunt de variables

Tot i que els resultats siguin quasi iguals, algunes mètriques empitjoren lleugerament, indicant que es possible que la variable *area* tot i aportar molta informació del soroll en les gràfiques hi ha casos puntuals els quals no es capaç de detectar.

Tercer conjunt de variables

En aquest tercer conjunt es va decidir complementar, altre cop, el conjunt de les gràfiques que han d'estar al voltant del 0 amb 4 característiques noves per gràfica, aquestes eren:

- `high_lim_cross`: Límit superior el qual indicaria que la funció ha pres valors massa elevats.
- `low_lim_cross`: Mateixa funció que el límit superior però per valors massa petits.
- `high_inner_lim_cross`: Límit superior interior, capta valors de menys magnitud que el límit superior, però en cas de que es creués masses cops aquest límit seria una senyal de perill.
- `low_inner_lim_cross`: Mateixa funció que el límit superior intern pero per valors petits.

A continuació es mostra una representació gràfica d'aquestes noves variables aplicades a dos senyals del mateix canal i del mateix assaig, però un assaig està etiquetat com a bo i l'altre com a dolent (realment els dos assaigs haurien d'estar etiquetats com a malament degut al soroll prolongat del senyal, però com ja s'ha comentat en altres apartats, la classificació que es disposa dels assaigs no és perfecte).

Aquest contingut ha estat retirat per confidencialitat

Figura 12: Variables sintètiques dels límits aplicades a una senyal amb soroll

En la il·lustració es mostren les variables mencionades com a:

- `high_lim_cross`: Màximo permitido
- `low_lim_cross`: Mínimo permitido
- `high_inner_lim_cross`: Máximo interno
- `low_inner_lim_cross`: Mínimo interno

Com bé s'ha mencionat tot i que un assaig estigui etiquetat com a bo (senyal: *Good*) i un altre com a dolent (senyal: *Bad*), ambdues senyals creuen repetidament els límits interiors, donant a entendre que son senyals massa sorolloses.
Cal destacar que no totes les gràfiques que pertanyen a aquest grup (dades al voltant del 0) actuen en la mateixa escala. Donat aquest fet, els valors dels quatre límits van estar fixats gràcies als enginyers del departament.

A continuació es mostra la distribució d'importàncies d'aquestes noves variables juntament amb el segon conjunt, formant el tercer conjunt.

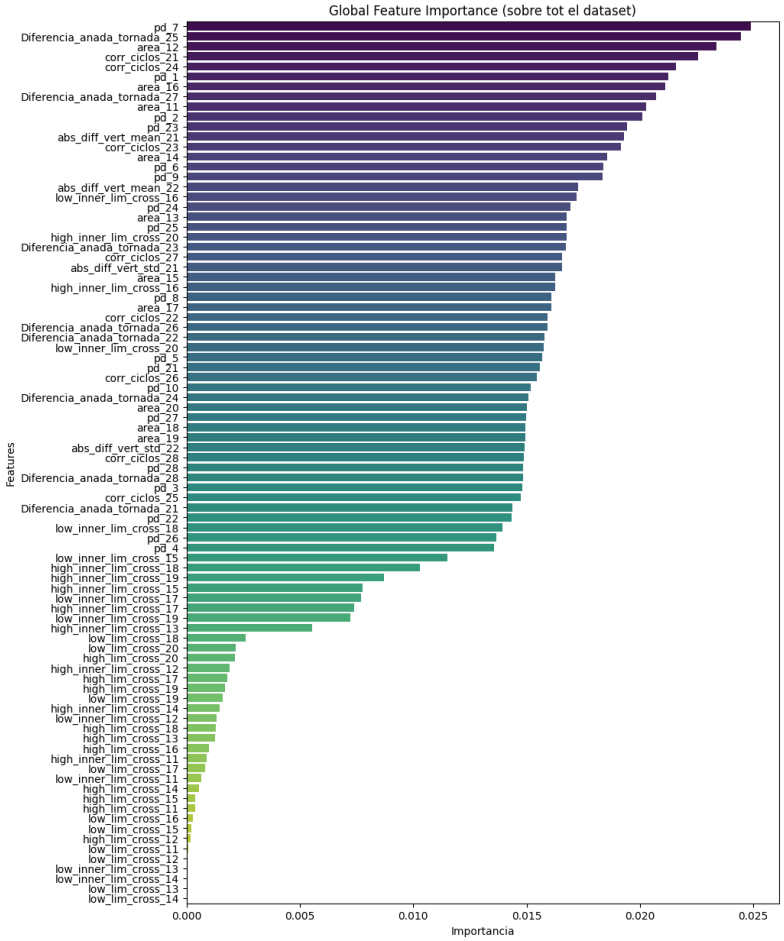

Figura 13: Importància del tercer conjunt de característiques en el model Random Forest

Seguidament es mostren les mètriques d'avaluació del model entrenat amb aquest tercer conjunt de variables sintètiques en comparació al conjunt principal.

Mètrica	**1er Conjunt**	**3n Conjunt**
Accuracy	0.7843	0.7866
Precision	0.6905	0.7
Recall	0.4085	0.4
F1-Score	0.5133	0.5091

Taula 7: Comparativa de mètriques entre conjunt de variables principal i tercer conjunt de variables

En aquest cas no es podria determinar quin dels dos conjunts de variables sintètiques dona millors resultats per al model només mirant la taula, ja que els resultats son molt semblants.
D'altre banda si s'observa el gràfic d'importàncies de les característiques, s'aprecia que n'hi ha moltes que contribueixen molt poc en el procès de classificació, donant a entendre que son variables prescindibles i és per aquest motiu, que el conjunt principal tot i comptar amb menys variables, aquestes en aporten més informació.

Quart conjunt

Per últim, es va decidir complementar el conjunt inicial amb noves variables que intentessin descriure més fidelment els altres dos grups de gràfiques: el grup de gràfiques amb comportament sinusoidal i suau i el grup de gràfiques específiques de l'assaig vertical. Aprofitant que aquests dos grups son conjunts, on normalment no hi ha soroll, es va plantejar la idea d'aplicar a totes les gràfiques d'aquests dos grups les següents variables estadístiques com a noves característiques:

- **Mean (Mitjana de la distribució)**: Mesura del valor mitjà d'una senyal al llarg del temps.
- **Skewness (Asimetria)**: Quantifica la simetria de la distribució. Una skewness positiva indica que la distribució té una cua més llarga a la dreta; negativa, cap a l'esquerra.
- **Kurtosis (Curtosi)**: Mesura la concentració de les dades al voltant de la mitjana. Valors elevats indiquen pics més aguts i cues més pesades.

A més, les mateixes variables s'han aplicat a les funcions derivades de les senyals, obtenint així 6 noves característiques per cada gràfica de comportament sinusoidal i específica de l'assaig.
A continuació es mostra el gràfic de les importàncies de totes les variables d'aquest últim conjunt.

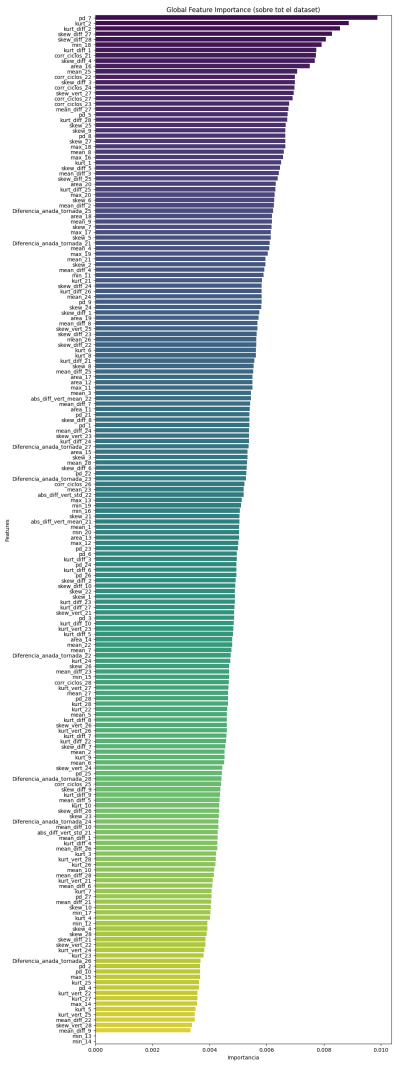

Figura 14: Importància del quart conjunt de característiques en el model Random Forest

Seguidament es mostra la taula comparativa de les mètriques d'avaluació d'aquest últim conjunt respecte al primer.

Mètrica	**1er Conjunt**	**4rt Conjunt**
Accuracy	0.7843	0.7708
Precision	0.6905	0.6875
Recall	0.4085	0.3143
F1-Score	0.5133	0.4314

Taula 8: Comparativa de mètriques entre conjunt de variables principal i quart conjunt de variables

Només tenint en compte el gràfic, es podria arribar a la conclusió de que totes les característiques que contribueixen en la presa de decisions del model. Malgrat la bona distribució de importàncies que presenta aquest últim conjunt, es pot observar en la taula anterior que, les mètriques important per al problema tractat, *Racall* i *F1-Score*, donen pitjors resultats que en els altres casos.
Aquests resultats indiquen que tot i que les característiques afegides siguin variables de caràcter estadístic enfocades a aportar més informació sobre els senyals, en moltes ocasions no aporten gaire valor. Un dels principals motius pels quals això pot passar, es que un vehicle pot tenir un comportament diferent a un altre en certes gràfiques, introduint variabilitat en les dades i confonent al model en la presa de decisions.

5.2 Corba ROC i AUC

A continuació es presenta la corba ROC (Receiver Operating Characteristic) [10] obtinguda durant la validació del model, juntament amb l'àrea sota la corba (AUC), que indica la capacitat del model per distingir entre classes positives i negatives.

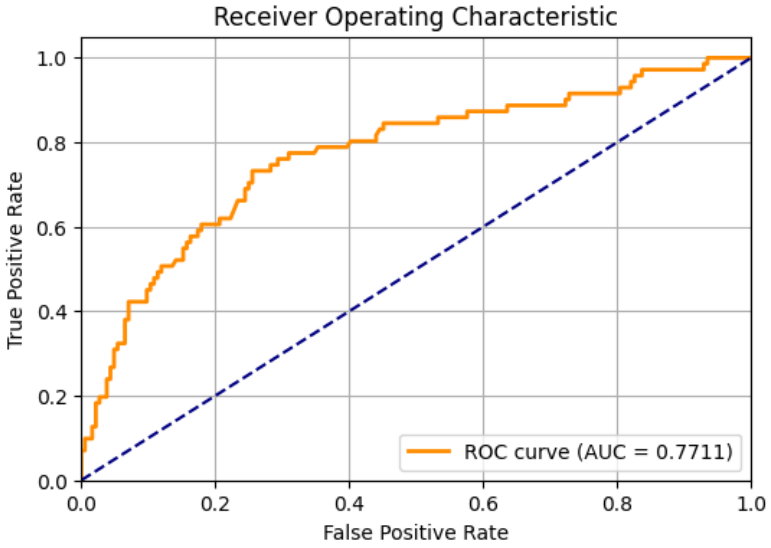

Figura 15: Corba ROC del model Random Forest

L'AUC obtingut és de **0.7711**, cosa que reflecteix una capacitat de discriminació moderadament alta.

Aquest valor d'AUC superior a 0.7 es pot considerar satisfactori en contextos reals, sobretot en entorns industrials, on s'accepta un cert marge d'error si el model manté una classificació robusta. Tot i que en el nostre cas la classificació no es prou informativa tenint en compte que hi ha més dades etiquetades com a vàlides que no pas el contrari, per tant, disposar d'un conjunt no balancejat de dades etiquetades fa que els casos vàlids predits com a vàlids contribueixin força més que els casos dolents predits com a dolents. Però s'ha de recordar que l'objectiu és classificar adequadament els casos erronis.

Tot i que el model mostra un bon rendiment general, aquest ha estat obtingut sense validació creuada, i per tant és convenient validar la seva estabilitat mitjançant validació creuada. Això permet detectar si hi ha un possible **sobreajustament** (overfitting) i si el comportament observat es manté en particions diferents de les dades. En aquest sentit, es proposa en apartats posteriors realitzar una avaluació més rigorosa mitjançant validació creuada i comparació amb altres models supervisats.

5.3 Matriu de Confusió sobre el conjunt d'entrenament

Per determinar si el model realment aprèn bé del conjunt d'entrenament, es va generar la matriu de confusió, però aquesta vegada pel conjunt d'entrenament.

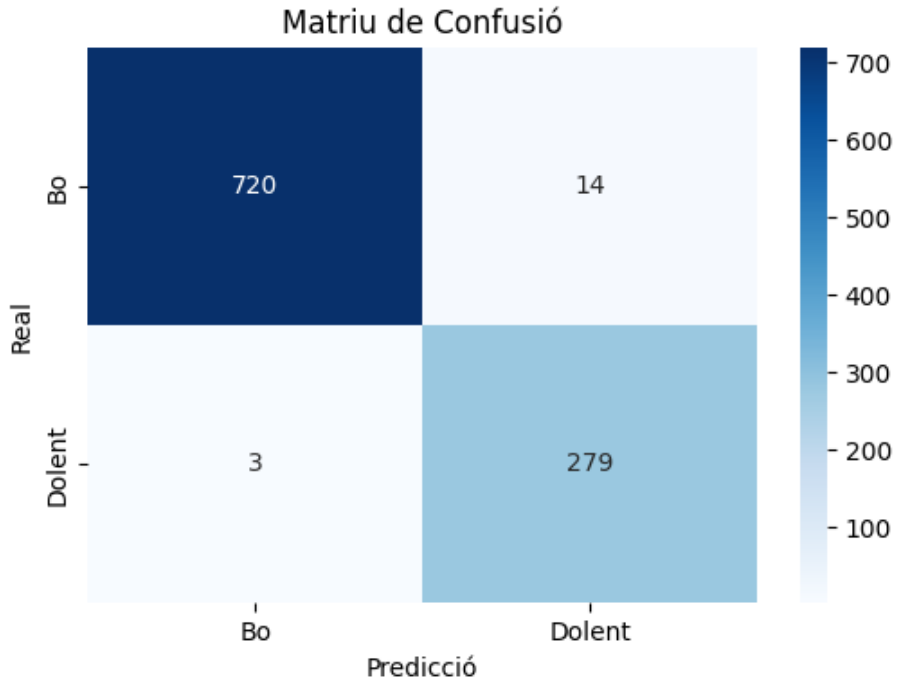

Figura 16: Matriu de confusió sobre el conjunt d'entrenament

Mètrica	Valors Train Set
Accuracy	0.9833
Precision	0.9522
Recall	0.9894
F1-Score	0.9833

Taula 9: Resultats de classificació del model Random Forest amb el conjunt d'entrenament

L'anàlisi d'aquesta matriu mostra una precisió gairebé perfecta, però també evidencia una diferència notable amb els resultats del conjunt d'entrenament, fet que suggereix la presència de sobreajustament, és a dir el model aprèn massa bé les dades d'entrenament, però generalitza pitjor sobre dades noves. En altres paraules es podria dir que el model en comptes d'aprendre patrons de classificació útils per conjunts nous, el que fa és memoritzar les dades d'entrenament.

Davant aquesta situació on el model presenta sobreajustament, s'han explorat diferents estratègies per reduir aquest efecte i millorar la capacitat de generalització del model:

- Ajust dels hiperparàmetres del Random Forest (nombre d'arbres, profunditat màxima, nombre de característiques seleccionades, etc.) mitjançant validació creuada.
- Entrenament i comparació de resultats d'altres models supervisats més avançats o no basats en arbres de decisió com ara: `XGBoost`, `LightGBM`, `Logistic Regression`, `Support Vector Machine (SVM)`, `K-Nearest Neighbours`, i`Multi Layer Perceptron`. Entrenament i validació de tots els models mitjançant validació creuada, per obtenir mètriques més estables i comparables.

6 Comparació amb Altres Models Supervisats

Un cop identificada la limitació del Random Forest, es va procedir a entrenar i comparar els resultats obtinguts amb altres models supervisats. Tots els models han estat ajustats amb els seus hiperparàmetres més rellevants i avaluats utilitzant el mateix conjunt de dades.

6.1 Estratègia d'optimització d'hiperparàmetres (Validació creuada)

5-Fold Stratified Cross-Validation

Per tal de validar els models de manera robusta i intentar evitar sobreajustaments (*overfitting*), s'ha optat per aplicar la tècnica de **5-Fold Stratified Cross- Validation** [18].

Aquesta metodologia consisteix en dividir el conjunt de dades en 5 particions (*folds*), de manera que, en cada iteració 4 particions de les particions s'utilitzen per entrenar el model i la partició restant s'utilitza per validar-lo. Aquest procés es repeteix 5 vegades, de manera que cada partició actua una vegada com a conjunt de validació. Per cada iteració es calcula una mètrica d'avaluació del model, en el nostre cas interessa la *f1 score*, i al final el valor que doni el model serà la mitjana dels valors donats en les iteracions.

$$F1 - Score_{Model} = \frac{1}{5}\sum_{i=1}^{5} F1 - Score_i \quad \text{on } F1 - Score_i \text{ representa la } \textit{F1-Score} \text{ de la } i\text{-èssima iteriació}$$

A continuació es mostra una referència d'aquest concepte.

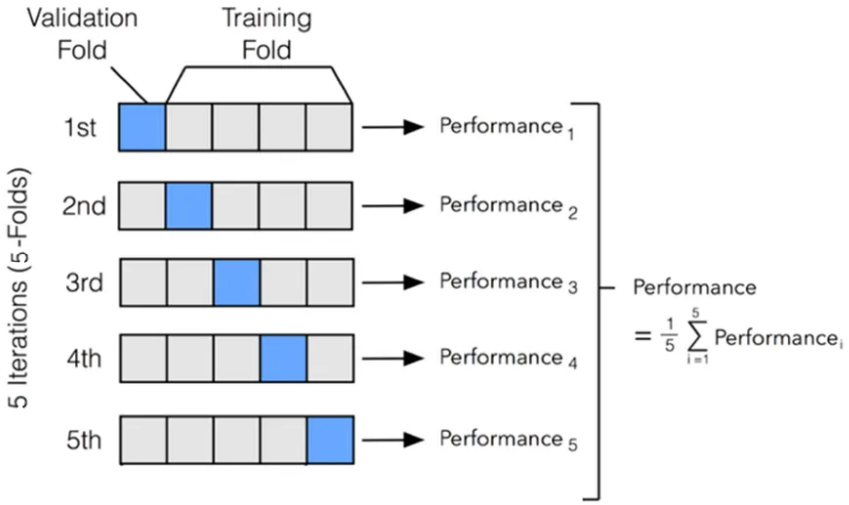

Figura 17: Metodologia `5-Fold Cross Validation`

En la il·lustració anterior els paràmetres *Performance*, correspondrien, en el nostre cas, al paràmetre *F1-Score*.

L'aspecte **stratified** garanteix que la proporció de classes dins de cada partició es manté similar a la del conjunt original, fet especialment rellevant en aquest cas, on el conjunt de dades mostra un cert **desequilibri entre classes** (amb més mostres etiquetades com a "bones" que "dolentes"). Aquest mecanisme assegura una millor representació de la realitat en cada subdivisió i millora la generalització del model.

Selecció de RandomizedSearchCV vs GridSearchCV

Per a la cerca dels millors hiperparàmetres de cada model hi ha diversos mètodes possibles, entre ells: **RandomizedSearchCV** i **GridSearchCV**.

- **GridSearchCV** prova totes les combinacions possibles d'un conjunt discret de valors, cercant de forma exhaustiva la combinació òptima.
- **RandomizedSearchCV**, en canvi, selecciona de forma aleatòria un nombre definit (`n_iter`) de combinacions de paràmetres dins l'espai possible.

En aquest context s'ha decidit emprar *RandomizedSearchCv* ja que, donat que molts models (com ara XGBoost, LightGBM o MLP) tenen un nombre elevat d'hiperparàmetres i possibles valors, es considera més adient **explorar més espai de paràmetres de forma eficient**, en comptes de buscar una única combinació òptima dins un espai més petit i discret.
A més, la utilització de RandomizedSearchCV redueix significativament el cost computacional i ofereix una bona aproximació al rendiment òptim del model, especialment quan es busca maximitzar mètriques com el *F1-score*, que és clau en aquest projecte.

Cal destacar que l'estratègia de `5-fold Stratified Cross-Validation` s'aplica per a cadascuna de les combinacions d'hiperparàmetres generades per `RandomizedSearchCV`. En aquest cas, com que s'ha definit `n_iter=30`, es proven 30 combinacions diferents, i per a cadascuna s'executen 5 iteracions de validació creuada. El millor model serà aquell que presenti la millor mitjana de puntuacions (en aquest cas, F1 Score) al llarg de les 5 particions del conjunt de dades.

6.2 Random Forest

El model de Random Forest [8] s'ha explicat en detall a la secció 4.1.2.

6.2.1 Validació creuada i resultats

Per tal de millorar el rendiment del model `Random Forest` i reduir possibles problemes de sobreajustament, es va aplicar un procés d'optimització d'hiperparàmetres mitjançant validació creuada. El mètode escollit fou una **validació creuada estratificada de 5 particions** (*5-fold Stratified Cross-Validation*), que assegura que cada partició mantingui la mateixa proporció de classes que el conjunt de dades original, fet especialment rellevant quan es treballa amb dades desbalancejades.

Per a l'optimització es va utilitzar la tècnica `RandomizedSearchCV`, que prova combinacions aleatòries d'hiperparàmetres dins d'un espai predefinit, fent-la més eficient computacionalment que una cerca exhaustiva (`GridSearchCV`) quan el nombre de combinacions és molt elevat, ja que en aquest cas resultava més adient tenir més combinacions de paràmetres que no buscar-ne la més òptima de totes. Els hiperparàmetres explorats van ser els següents:

- `n_estimators`: nombre d'arbres del bosc (entre 100 i 300),
- `max_depth`: profunditat màxima dels arbres (entre 2 i 30),
- `min_samples_split`: nombre mínim de mostres per dividir un node intern (entre 2 i 30),
- `min_samples_leaf`: nombre mínim de mostres a una fulla (entre 2 i 30),
- `max_features`: nombre de característiques considerades en cada divisió (opcions clàssiques com `'sqrt'` o `'log2'`, així com valors enters específics).

L'objectiu d'optimització va ser la mètrica `F1-Score`, que equilibra la precisió i la sensibilitat, fet rellevant en entorns on detectar la classe positiva (casos de fallada o anomalia) és crític.

Els millors hiperparàmetres obtinguts durant el procés van ser:

```
{'max_depth': 26, 'max_features': None, 'min_samples_leaf': 6, 'min_samples_split':
                                20, 'n_estimators': 234}
```

Aquest conjunt d'hiperparàmetres indica un model amb arbres relativament profunds, s'exigeix un nombre mínim de 6 mostres per a crear fulles i 20 per dividir un node, cosa que redueix el risc de sobreajustament. A més, l'opció `max_features = None` permet utilitzar totes les característiques en cada divisió, donant més flexibilitat a cada arbre individual, tot i que pot augmentar la variància del model.

Els resultats sobre el conjunt de test utilitzant aquest model ajustat foren els següents:

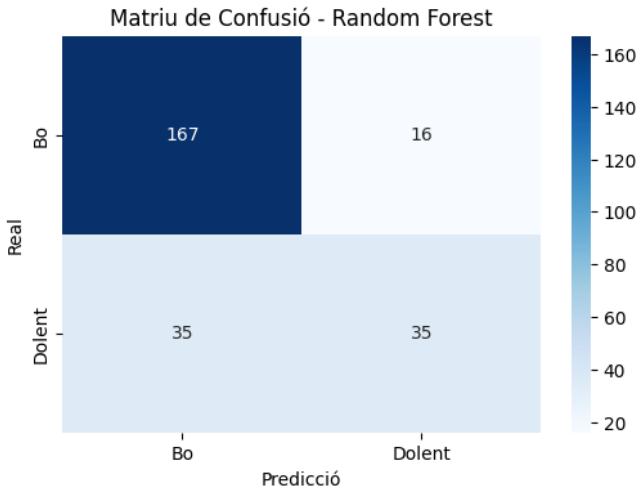

Figura 18: Matriu de confusió del model `Random Forest` amb Validació creuada

Mètrica	**Valors**
Accuracy	0.7984
Precision	0.6863
Recall	0.5000
F1-Score	0.5785

Taula 10: Resultats de classificació del model Random Forest amb el mètode *Cross Validation*

Tot i que la precisió és força elevada, la sensibilitat és moderada, suggerint que encara hi ha casos positius que no són detectats. A continuació es farà una comparació entre les mètriques aconseguides pel primer model de *Random Forest* entrenat, juntament amb les del mateix model però amb el conjunt d'entrenament per validar-lo i amb aquest últim model de *Random Forest* amb el mètode d'ajustament de paràmetres *Cross Validation*.

Seguidament es mostra la taula de comparació de les mètriques en els tres casos diferents.

Mètrica	RF	RF CV	RF Train Set
Accuracy	0.7843	0.7984	0.9833
Precision	0.6905	0.6863	0.9522
Recall	0.4085	0.5000	0.9894
F1-Score	0.5133	0.5785	0.9833

Taula 11: Resultats de classificació del model Random Forest amb i sense el mètode *Cross Validation* i amb el conjunt d'entrenament

On *RF* indica el mdoel *Random Forest* inicial, *RF CV* indica el mateix model però amb ajustament dels seus paràmetres i *RF Train Set* fa referència al primer model però mirant l'avaluació mitjançant el conjunt d'entrenament.
A continuació es mostra una visualització de es anteriors dades en forma d'il·lustració.

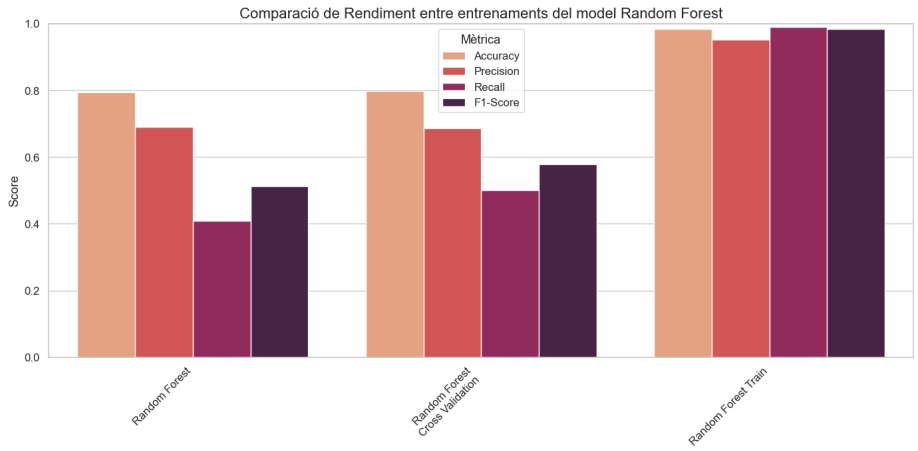

Figura 19: gràfica comparativa del rendiment del model Random Forest en diferents situcaicons

Com es pot veure en la gràfica tot i que els valors de la sensibilitat i *F1-Score* han augmentat respecte el model sense l'ajustament dels paràmetres, aquests continuen estan lluny de les mètriques sobre el conjunt l'entrenament. Indicant que el fenomen de sobreajustament no prové de l'ajustament dels paràmetres del model.

6.3 XGBoost

A diferència de `Random Forest`, on els arbres es generen de forma independent i els resultats s'agreguen mitjançant votació, el `XGBoost` construeix arbres de manera seqüencial. Cada arbre nou intenta corregir els errors del conjunt d'arbres anteriors ajustant-se als residus de les prediccions prèvies. Això permet aprendre de manera més específica patrons que els models previs no han pogut capturar.

XGBoost incorpora diversos mecanismes que milloren notablement el rendiment computacional i la capacitat de generalització del model:

- Utilitza estructures internes optimitzades per a la construcció d'arbres (*approximate tree learning*).
- Permet el tractament directe de valors nuls.
- Inclou opcions per a la paral·lelització de càlculs durant l'entrenament.
- Permet l'ús de *early stopping* i validació creuada de manera nativa.

Tanmateix, XGBoost no és un model trivial de configurar. Té nombrosos hiperparàmetres (com `learning_rate`, `max_depth`, `subsample`, `colsample_bytree`, etc.) que cal ajustar acuradament mitjançant tècniques de validació creuada per tal d'obtenir un rendiment òptim. A més, pot ser sensible al soroll en les dades i als valors extrems si no es preprocesa adequadament la informació.

En el cas del problema tractat en aquest treball, XGBoost és especialment adequat gràcies a la seva capacitat per modelar relacions no lineals i interactives entre variables. La seva estratègia d'entrenament seqüencial li permet adaptar-se bé a patrons subtils que poden ser indicatius d'un mal funcionament en els assaigs, cosa que el converteix en una opció potent per a la classificació binària.

6.3.1 Validació creuada i resultats

Per al model `XGBoost`, s'ha optat també per fer ús de la tècnica `RandomizedSearchCV` amb una validació creuada estratificada de 5 particions (*5-fold Stratified Cross-Validation*). Degut al nombre elevat d'hiperparàmetres i a la seva naturalesa contínua en molts casos (com ara `learning_rate`, `colsample_bytree` o `subsample`), resultava més eficient explorar una gran varietat de combinacions aleatòries dins de rangs amplis, en comptes d'intentar avaluar totes les opcions possibles. Aquesta decisió permet obtenir bons resultats amb un cost computacional més raonable.

Els hiperparàmetres considerats en la cerca han inclòs:

- `n_estimators`: nombre d'arbres (entre 100 i 300),
- `max_depth`: profunditat màxima de cada arbre (entre 2 i 30),
- `learning_rate`: taxa d'aprenentatge (entre 0.01 i 0.31),
- `subsample` i `colsample_bytree`: proporcions d'instàncies i característiques utilitzades per arbre (valors continus entre 0.05 i 1),
- `min_child_weight`: nombre mínim d'instàncies per node fulla,
- `gamma`: guany mínim per realitzar una partició addicional,
- `scale_pos_weight`: pes relatiu de la classe minoritària per fer front a desequilibris de classe.

Els millors paràmetres trobats han estat:

```
{colsample_bytree = 0.5092, gamma = 0.2336, learning_rate = 0.0529, max_depth =
4, min_child_weight = 2, n_estimators = 187, scale_pos_weight = 2.6189, subsample
= 0.9886}
```

Aquests valors mostren una configuració conservadora però ben ajustada: una profunditat d'arbre moderada (`max_depth = 4`) que evita sobreajust, una taxa d'aprenentatge baixa (`learning_rate` $\approx$ 0.05) per assegurar un entrenament estable, i un nombre d'arbres (`n_estimators = 187`) suficient per capturar patrons rellevants. La combinació de valors de `subsample` i `colsample_bytree`, properes però no iguals a 1, introdueix aleatorietat que millora la capacitat de generalització.

Quant al rendiment sobre el conjunt de test, els resultats obtinguts han estat:

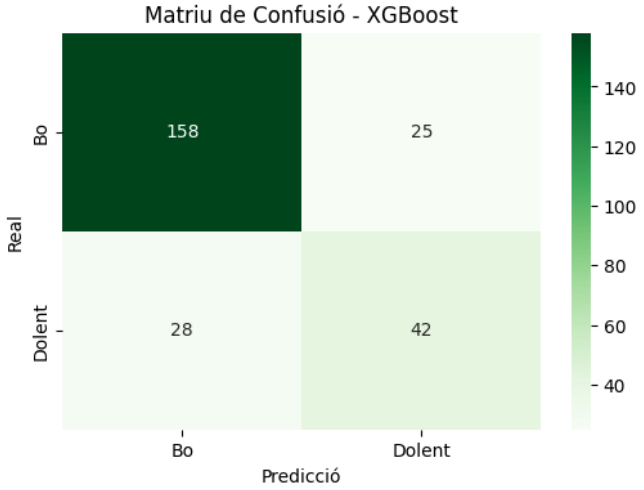

Figura 20: Matriu de confusió del model `XGBoost` amb Validació creuada

Mètrica	Valors
Accuracy	0.7905
Precision	0.6269
Recall	0.6000
F1-Score	0.6131

Taula 12: Resultats de classificació del model XGBoost

Aquestes mètriques reflecteixen un equilibri raonable entre detecció i precisió, especialment en un context amb certa asimetria entre classes. El valor elevat de `scale_pos_weight` mostra que el model ha après a compensar aquesta desproporció, ajustant-se millor al comportament de les classes minoritàries.

El codi de l'entrenament es trobarà al punt 9.7.4 de l'annex.

6.4 LightGBM

Com en el cas anterior, el model `LightGBM` construeix arbres de decisió seqüencialment, on cada arbre intenta corregir els errors dels arbres anteriors. Tanmateix, introdueix dues innovacions clau que el distingeixen d'altres mètodes:

1. **Crescuda basada en fulles (leaf-wise growth)**: En lloc d'expandir els arbres nivell a nivell (com fa XGBoost amb estratègia *level-wise*), LightGBM afegeix noves particions al full que més redueix l'error. Això dona lloc a arbres més profunds però més eficients a l'hora de reduir la pèrdua.
2. **Histogram-based decision trees**: Les dades contínues es discretitzen en bins, cosa que redueix el nombre de comparacions necessàries durant la construcció de l'arbre, millorant considerablement la velocitat i l'ús de memòria.

Aquest enfocament leaf-wise, tot i ser més eficient en termes de rendiment, pot augmentar el risc de sobreajustament si no es controla la profunditat dels arbres (`max_depth`) o altres paràmetres com `min_child_samples`. És per això que LightGBM incorpora múltiples opcions de regularització i control de la complexitat del model, com:

- `num_leaves`: controla el nombre màxim de fulles, limitant la complexitat dels arbres.
- `min_data_in_leaf` i `min_child_samples`: eviten que el model s'ajusti a mostres molt petites.
- `feature_fraction` i `bagging_fraction`: seleccionen aleatòriament subconjunts de variables o instàncies per a cada arbre, afavorint la generalització.

A més, LightGBM pot gestionar eficientment valors nuls i columnes amb moltes categories (*categorical features*), ja que internament implementa estratègies específiques per determinar l'ordre òptim de particions d'aquestes variables.

Tot i això, l'alta capacitat expressiva del model també el fa més susceptible al sobreajustament si no s'aplica una validació creuada rigorosa i una selecció acurada d'hiperparàmetres. En aquest estudi, s'ha utilitzat validació creauda per ajustar el nombre òptim de fulles, el percentatge de mostreig i altres paràmetres que permeten obtenir un bon equilibri entre complexitat i generalització.

6.4.1 Validació creuada i resultats

LightGBM és un model especialment sensible a paràmetres com `num_leaves` o `min_child_samples`, i optimitzar aquests valors pot tenir un impacte molt significatiu en el rendiment final del model. En aquest cas, es va limitar la cerca a 30 combinacions aleatòries per equilibrar cost computacional i cobertura d'espai.

A més, s'han tingut en compte els desequilibris de classe presents en el conjunt de dades i s'han incorporat pesos específics per classe mitjançant la utilitat `compute_class_weight`, integrant-los dins del paràmetre `class_weight` del model base. Aquesta mesura ajuda a penalitzar adequadament les errades en la classe minoritària, millorant l'exhaustivitat o sensibilitat (*recall*) i l'equilibri global del model.

Els hiperparàmetres explorats incloïen:

- `n_estimators`: nombre d'arbres totals,
- `max_depth`: profunditat màxima dels arbres (amb -1 per a creixement il·limitat),
- `learning_rate`: taxa d'aprenentatge,

- `num_leaves`: nombre màxim de fulles per arbre, clau per a la complexitat,
- `min_child_samples`: nombre mínim d'instàncies per a fer una partició,
- `subsample` i `colsample_bytree`: proporcions d'instàncies i característiques utilitzades per arbre.

Els millors paràmetres trobats durant el procés de cerca van ser:

```
{colsample_bytree = 0.9687, learning_rate = 0.0268, max_depth = 8, min_child_samples
= 30, n_estimators = 181, num_leaves = 125, subsample = 0.9315}
```

Aquest conjunt de valors mostra un model equilibrat i altament ajustat. La profunditat moderada dels arbres (`max_depth = 8`) juntament amb un nombre elevat de fulles per arbre (`num_leaves = 125`) indica una configuració prou flexible per capturar patrons complexos, però amb control per evitar sobreajust. La taxa d'aprenentatge baixa (`learning_rate` $\approx$ `0.027`) reforça aquest comportament gradual i estable. Els valors elevats de `subsample` i `colsample_bytree` contribueixen a la diversificació dels arbres, millorar la generalització i prevenir el sobreajustament.

Els resultats obtinguts sobre el conjunt de validació són els següents:

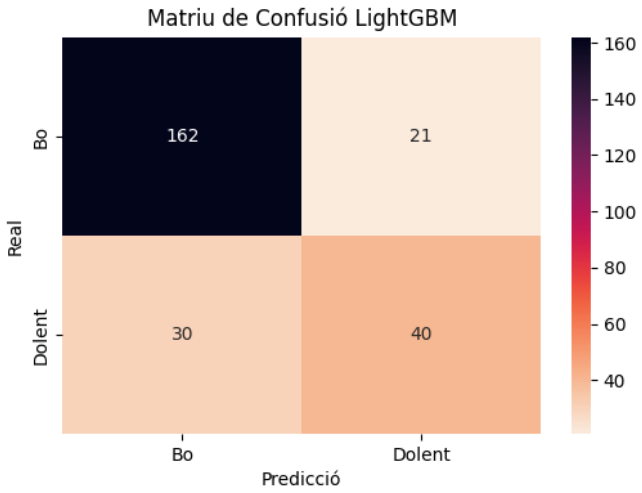

Figura 21: Matriu de confusió del model `LightGBM` amb Validació creuada

Mètrica	Valors
Accuracy	0.7984
Precision	0.6557
Recall	0.5714
F1-Score	0.6107

Taula 13: Resultats de classificació del model LightGBM

Aquestes mètriques situen el model LightGBM en una posició competitiva respecte altres models basats en arbres, amb una bona relació entre detecció de positius i fiabilitat de les prediccions. La seva eficiència computacional i flexibilitat el fan especialment atractiu per a aquest tipus de problemes estructurats.

El codi de l'entrenament es trobarà al punt 9.7.5 de l'annex.

6.5 Regressió Logística

La `Regressió Logística` [13] és un model de classificació supervisada fonamentat en estadística clàssica. Tot i el seu nom, no es tracta d'un mètode de regressió en sentit estricte, sinó d'un model discriminatiu que estima la probabilitat que una observació pertanyi a una determinada classe binària (etiquetada com 0 o 1).

El model s'ajusta, és a dir, s'identifiquen els millors coeficients, mitjançant el mètode de **Màxima Versemblança** (*Maximum Likelihood Estimation*, MLE).
Sigui $\{(x^\mu, z^\mu)\}_{\mu=1,\dots,p}$ un conjunt de mostres, on $z^\mu \in \{0, 1\}$ indica la classe. La probabilitat que $z^\mu = 1$ ve donada per:

$$p_\mu \equiv p(x^\mu) = \frac{1}{1 + e^{-(a \cdot x^\mu + b)}}$$

Versemblança: és la probabilitat d'observar les variables dependents de la mostra donats els coeficients actuals (és a dir, *la probabilitat d'observar les dades donat el model actual*).
La funció de versemblança que es vol maximitzar és:

$$L(a, b) = \prod_\mu (z^\mu p_\mu + (1 - z^\mu)(1 - p_\mu))$$

La regressió logística assumeix que la relació entre les característiques i la **logit** de la probabilitat és lineal, cosa que pot ser limitant en casos on les fronteres de decisió siguin no lineals. A més, és sensible a variables correlacionades o molt dispars en escala, de manera que sol requerir prèviament una normalització de les dades i, sovint, una selecció de característiques per millorar el rendiment.

En el context d'aquest estudi, s'ha aplicat la regressió logística per detectar si les diferències entre les dues classes poden ser capturades per una combinació lineal de les característiques. Si bé no s'espera que aquest model superi mètodes més complexos com Random Forest o XGBoost, serveix com a punt de referència bàsic i interpretable per entendre l'estructura de les dades i validar la rellevància de les variables.

6.5.1 Validació creuada i resultats

Tot i ser un dels models més senzills en el camp de la classificació, la Regressió Logística sovint pot oferir bons resultats quan les relacions entre variables són predominantment lineals. L'objectiu ha estat ajustar el paràmetre de regularització C, el qual controla la força de la penalització: valors grans corresponen a menys regularització (és a dir, més flexibilitat per ajustar-se al conjunt d'entrenament).

L'espai d'hiperparàmetres incloïa:

- `C`: valor del paràmetre d'inversió de la regularització, seleccionat aleatòriament en l'interval (0.01, 300),
- `penalty`: s'ha fixat a `l2`, que correspon a regularització Ridge,
- `solver`: s'ha utilitzat l'optimitzador `lbfgs`, adequat per models amb moltes observacions i regularització `l2`.

Cal destacar que el conjunt d'entrenament s'ha normalitzat (`X_train_scaled`) abans de l'entrenament, ja que la regressió logística pot ser sensible a l'escala de les característiques. La normalització és especialment rellevant quan s'utilitzen regularitzacions, ja que aquestes penalitzen pesos més grans.

Els millors hiperparàmetres trobats han estat:

```
{C = 177.73, penalty = 'l2', solver = 'lbfgs'}
```

Aquest valor elevat de C indica una regularització molt feble, és a dir, un model que ha estat permès ajustar-se gairebé lliurement al conjunt d'entrenament. Això podria haver estat problemàtic en un context amb desequilibri de classes, ja que el model pot no generalitzar bé.

Els resultats obtinguts han estat:

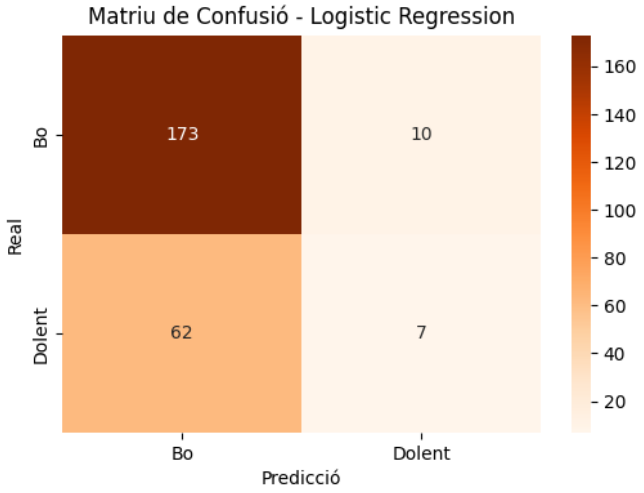

Figura 22: Matriu de confusió del model `Logistic Regression` amb Validació creuada

Mètrica	Valors
Accuracy	0.7143
Precision	0.4211
Recall	0.1159
F1-Score	0.1818

Taula 14: Resultats de classificació del model Logistic Regression

Com es pot observar, el model mostra un comportament significativament inferior respecte als models basats en arbres. Tot i tenir una precisió moderada, la sensibilitat és molt baixa, indicant que només una petita fracció de les instàncies positives han estat detectades. Aquest fet provoca una puntuació F1 molt reduïda, que evidencia el mal ajust del model a les característiques del problema. En un escenari com aquest, amb relacions no necessàriament lineals i amb desequilibri de classes, és esperable que la Regressió Logística no sigui competitiva.

El codi de l'entrenament es trobarà al punt 9.7.6 de l'annex.

6.6 Support Vector Machines (SVM)

El model `Support Vector Machines` (SVM) [9] és una tècnica de classificació supervisada que cerca trobar la millor frontera de decisió que separa dues classes. Aquesta frontera, o hiperplà, es defineix de manera que la distància entre els punts més propers de cada classe (anomenats *support vectors*) i l'hiperplà sigui màxima. Aquest enfocament es coneix com a *marge màxim*, i és fonamental per a la seva robustesa en la generalització.

Formalment, si es consideren dades etiquetades $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, amb $\boldsymbol{x}_i \in \mathbb{R}^d$ i $y_i \in \{-1, 1\}$, l'objectiu de l'SVM lineal és trobar un vector $\boldsymbol{w}$ i un biaix b que satisfacin:

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 \quad \text{sota la restricció} \quad y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1$$

Aquest problema es pot resoldre mitjançant tècniques d'optimització com multiplicadors de *Lagrange*. En la pràctica, moltes dades no són linealment separables, per això es permeten certes violacions del marge mitjançant el paràmetre de regularització C, que controla el compromís entre un marge ample i la penalització dels errors de classificació.

Una de les grans fortaleses de l'SVM és la seva capacitat per manejar problemes no lineals mitjançant l'ús de **nuclis** (*kernels*). A través del truc del nucli, les dades es projecten implícitament en espais de major dimensió on poden ser separables linealment. Els nuclis més comuns inclouen:

- **RBF (Radial Basis Function)**: $\exp(-\gamma\|\boldsymbol{x} - \boldsymbol{x}'\|^2)$
- **Polinòmic**: $(\boldsymbol{x}^\top \boldsymbol{x}' + c)^d$
- **Sigmoide**: $\tanh(\kappa \boldsymbol{x}^\top \boldsymbol{x}' + \theta)$

Els punts forts del model SVM inclouen la seva eficàcia en espais d'alta dimensió i la seva robustesa davant problemes on el nombre de mostres és reduït respecte al nombre de característiques. A més, l'ús de nuclis li permet capturar relacions complexes entre variables.

No obstant això, presenta algunes limitacions. L'entrenament pot ser costós computacionalment per a conjunts de dades molt grans, especialment amb nuclis no lineals. A més, la seva interpretabilitat és baixa en comparació amb models com la regressió logística, i la selecció dels hiperparàmetres (com C i γ) pot tenir un impacte crític en el rendiment final.

En el context d'aquest estudi, l'SVM ofereix un enfocament alternatiu als models basats en arbres. Tot i que pot no captar de forma òptima patrons molt estructurats com ho fan Random Forest o XGBoost, és especialment útil per a problemes on les diferències entre classes estan distribuïdes en una superfície de decisió clara però no necessàriament lineal. El seu rendiment dependrà fortament de l'elecció del nucli i la calibració dels paràmetres.

6.6.1 Validació creuada i resultats

Aquest tipus de model és potent en espais de característiques d'elevada dimensionalitat i pot ser especialment eficaç quan les classes són clarament separables, encara que sigui de manera no lineal.

El model base ha estat inicialitzat amb l'opció `probability=True` per tal de poder obtenir prediccions probabilístiques, i s'han explorat els següents hiperparàmetres:

- `C`: coeficient de regularització, mostrejat uniformement entre 0.1 i 10. Valors alts impliquen menys tolerància a l'error de classificació.
- `gamma`: controla l'amplitud del nucli (per als nuclis no lineals). S'han provat les opcions `'scale'` i `'auto'`.

- `kernel`: tipus de funció nucli utilitzada per projectar les dades en espais de dimensionalitat superior. S'han provat els nuclis `'rbf'`, `'poly'` i `'sigmoid'`.

L'ús de `RandomizedSearchCV` amb 20 iteracions permet una exploració eficient de l'espai de cerca, ja que el cost computacional d'entrenar diversos models SVM pot ser considerable, especialment amb nuclis complexos com `rbf` o `poly`.

Els millors hiperparàmetres trobats han estat:

```
{C = 7.42, gamma = 'scale', kernel = 'rbf'}
```

L'elecció d'un nucli `rbf` (Radial Basis Function) és coherent amb la naturalesa potencialment no lineal del problema. El valor de `C` moderadament alt indica que el model ha estat configurat per penalitzar força els errors de classificació, intentant ajustar-se millor a les dades d'entrenament.

Pel que fa als resultats:

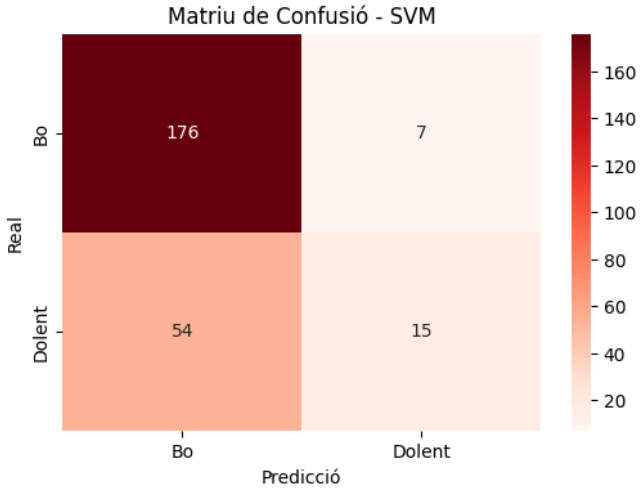

Figura 23: Matriu de confusió del model `Support Vector Machine` amb Validació creuada

Mètrica	Valors
Accuracy	0.7579
Precision	0.6818
Recall	0.2174
F1-Score	0.3297

Taula 15: Resultats de classificació del model Support Vector Machine

Tot i obtenir una precisió acceptable, el model mostra una capacitat molt limitada per detectar instàncies positives (sensibilitat molt baixa), i per tant, l'F1 Score es veu greument afectat. Aquest resultat pot indicar que, tot i la flexibilitat del nucli RBF, el model no és capaç de capturar adequadament la distribució de la classe minoritària. A més, la sensibilitat del SVM a la selecció del nucli i a l'escala de les característiques pot explicar el rendiment discret en comparació amb models més robustos davant desequilibris, com els basats en arbres.

El codi de l'entrenament es trobarà al punt 9.7.7 de l'annex.

6.7 K-Nearest Neighbours (KNN)

La lògica del `K-Nearest Neighbors` (KNN) [11] és senzilla: per a una nova mostra, el model cerca els k veïns més propers dins l'espai de característiques segons una mètrica de distància (habitualment, la distància Euclidiana) i assigna la classe més comuna entre aquests veïns. En classificació binària, aquest procés es pot veure com una votació majoritària entre els veïns seleccionats.

$$\hat{y} = \text{majority\_vote}\,\{y_i : x_i \in \mathcal{N}_k(x)\}$$

on $\mathcal{N}_k(x)$ representa el conjunt dels k veïns més propers a x.

L'elecció del valor de k és crítica: un valor petit pot fer el model sensible al soroll (sobreajustament), mentre que un valor massa gran pot suavitzar massa la frontera de decisió i provocar underfitting. A més, KNN assumeix que totes les característiques tenen el mateix pes en el càlcul de distància, per la qual cosa és imprescindible una bona normalització o estandarditzación de les dades.

Els punts forts de KNN són la seva simplicitat i interpretabilitat. Funciona bé en conjunts de dades petits o moderats i en problemes on les classes estan ben separades en l'espai de característiques. No requereix una fase de modelització complexa i s'adapta a fronteres de decisió de forma arbitrària, cosa que el fa flexible en contextos no lineals.

No obstant això, té diverses limitacions: la seva eficiència computacional és baixa en predicció per a conjunts de dades grans, ja que requereix calcular distàncies amb totes les instàncies d'entrenament. També és molt sensible a la dimensió de l'espai (la coneguda *curse of dimensionality*) i pot patir si hi ha moltes característiques no informatives.

Aplicat al problema estudiat, KNN ofereix una perspectiva no paramètrica i local, és a dir, basa les seves decisions exclusivament en l'estructura de veïnatge. Això pot ser útil en escenaris on existeixen agrupacions clares d'exemples similars, però pot fallar si les dades són molt sorolloses o si no existeix una estructura espacial clara entre les instàncies. Per aquest motiu, s'espera que el seu rendiment sigui competitiu però probablement inferior al de models més globals i estructurats com Random Forest o XGBoost.

6.7.1 Validació creuada i resultats

K-Nearest Neighbors (KNN): validació creuada i selecció d'hiperparàmetres

El model de *K-Nearest Neighbors* (KNN) és un classificador no paramètric que classifica una instància segons la majoria de les etiquetes dels seus k veïns més propers. La seva simplicitat i interpretabilitat el fan útil com a model base, tot i que pot tenir limitacions importants davant conjunts de dades amb soroll, dimensionalitat elevada o desequilibris entre classes.

Per tal de determinar els hiperparàmetres òptims, s'ha utilitzat l'espai següent d'hiperparàmetres:

- `n_neighbors`: nombre de veïns a considerar. Es mostreja aleatòriament entre 1 i 10.
- `weights`: manera de ponderar els veïns. Opcions: `'uniform'` (tots igual) o `'distance'` (els més propers pesen més).
- `metric`: mètrica de distància. Es comparen `'euclidean'` i `'manhattan'`.

Aquest enfocament permet explorar diferents combinacions que poden tenir efectes rellevants segons la distribució de les dades i la seva escala (d'aquí la importància d'haver utilitzat dades prèviament normalitzades).

Els millors hiperparàmetres trobats han estat:

```
{n_neighbors = 2, weights = 'distance', metric = 'euclidean'}
```

L'ús de només 2 veïns pot resultar en un model molt sensible al soroll local, però el fet d'utilitzar una ponderació per distància pot mitigar parcialment aquest efecte. La mètrica euclidiana és l'elecció habitual en espais on totes les característiques estan escalades, com és el cas.

Pel que fa al rendiment del model:

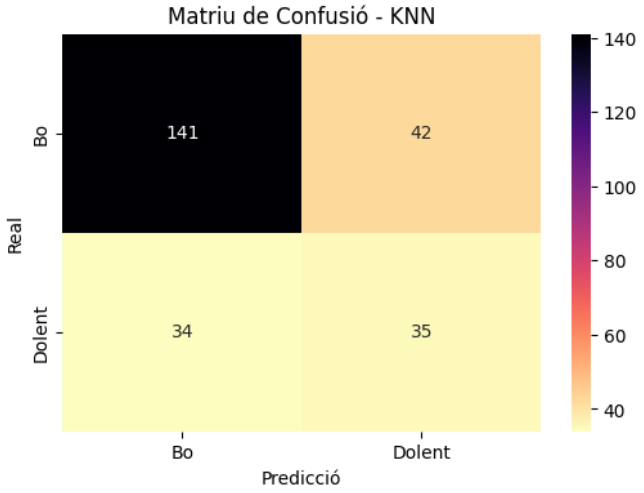

Figura 24: Matriu de confusió del model `K-Nearest Neighbours` amb Validació creuada

Mètrica	Valors
Accuracy	0.6984
Precision	0.4545
Recall	0.5072
F1-Score	0.4795

Taula 16: Resultats de classificació del model K-Nearest Neighbours

El model KNN mostra una capacitat de detecció de la classe minoritària (sensibilitat) superior a la d'altres models lineals, com la regressió logística. Tanmateix, els resultats no han estat òptims, a més pot veure's fortament afectat per canvis en la distribució local de les dades, fet que limita la seva robustesa en escenaris reals.

El codi de l'entrenament es trobarà al punt 9.7.8 de l'annex.

6.8 Multi-Layer Perceptron (MLP)

El `Multi-Layer Perceptron` (MLP) [17] és un model supervisat que consisteix en capes de neurones organitzades seqüencialment: una capa d'entrada, una o més capes ocultes i una capa de sortida. Cada neurona d'una capa està connectada amb totes les neurones de la següent capa, formant una arquitectura totalment connectada.

El funcionament d'un MLP es basa en la composició de transformacions lineals seguides per funcions d'activació no lineals. En concret, el càlcul d'una neurona és:

$$z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)}, \quad a^{(l)} = \phi(z^{(l)})$$

on $W^{(l)}$ i $b^{(l)}$ són els pesos i el biaix de la capa l, $a^{(l-1)}$ és la sortida de la capa anterior, i ϕ és una funció d'activació com ReLU, tanh o sigmoid. La sortida final s'obté després de propagar les dades cap endavant (*forward propagation*) i aplicar una funció d'activació a la darrera capa.

L'ajust dels pesos es fa mitjançant *backpropagation*, una tècnica d'optimització que calcula el gradient de l'error respecte als pesos i l'actualitza mitjançant descens de gradient (o alguna de les seves variants com Adam o RMSProp). Aquesta capacitat d'aprendre representacions jeràrquiques i no lineals fa que els MLP siguin molt potents per capturar patrons complexos.

Els punts forts del MLP inclouen:

- Capacitat per aprendre relacions no lineals i representar funcions molt complexes.
- Flexibilitat arquitectònica: el nombre de capes i neurones pot ajustar-se a la complexitat del problema.
- Versatilitat: pot aplicar-se a classificació binària, multiclasse i regressió.

Tanmateix, també presenta algunes debilitats:

- Alta dependència de l'escala de les dades: la normalització és essencial per al bon funcionament.
- Elevat risc de sobreajustament si no s'aplica regularització (com dropout o weight decay).
- Requereix més dades i càlcul que models més senzills per assolir bon rendiment.

En el context d'aquest treball, l'MLP representa una alternativa poderosa i teòricament molt flexible. Tanmateix, l'espai de dades és relativament limitat en mida i dimensionalitat, fet que pot dificultar que el MLP destaqui sobre altres models com XGBoost o Random Forest, que estan més ben adaptats a treballar amb conjunts de dades estructurades i tabulars. Tot i això, pot ser útil per explorar si existeixen interaccions complexes entre variables que altres models lineals o basats en arbres no capten bé.

6.8.1 Validació creuada i resultats

El *Multi-Layer Perceptron* (MLP) és un model de xarxa neuronal artificial de tipus *feedforward*, capaç de capturar relacions no lineals complexes entre les variables. Tot i això, el seu entrenament pot ser més lent i inestable en comparació amb altres models, especialment si no es tria adequadament l'estructura de la xarxa i els hiperparàmetres.

L'espai de cerca ha inclòs els següents paràmetres:

- `hidden_layer_sizes`: diferents configuracions de capes ocultes, incloent una i dues capes amb diverses combinacions (per exemple: (100,), (100, 100), (50, 100)).

- `activation`: funció d'activació, amb opcions entre `'relu'` i `'tanh'`.
- `solver`: optimitzador utilitzat, fixat a `'adam'`, que és robust per a conjunts de dades amb moltes característiques.
- `alpha`: terme de regularització L2 per evitar sobreajust, mostrejat aleatòriament entre 0.0001 i 0.001.
- `learning_rate`: estratègia d'ajust del ritme d'aprenentatge, amb opcions `'constant'` i `'adaptive'`.

Els millors hiperparàmetres trobats van ser:

```
{hidden_layer_sizes = (100, 100), activation = 'relu', alpha = 0.000265, solver
= 'adam', learning_rate = 'constant'}
```

L'estructura de dues capes amb 100 neurones cadascuna és habitual i raonable per a problemes amb un cert grau de complexitat no lineal. L'activació `'relu'` és la més utilitzada per la seva eficiència computacional i el seu bon comportament en pràctica. El valor d'α és petit però no nul, proporcionant una lleu regularització sense impedir la flexibilitat del model.

Pel que fa als resultats de rendiment:

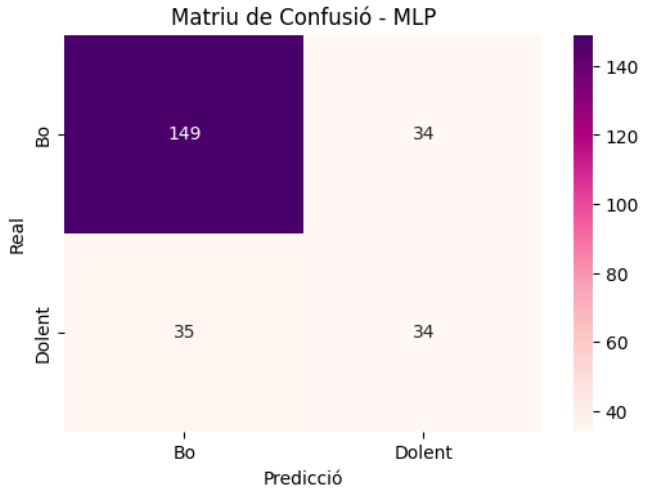

Figura 25: Matriu de confusió del model `Multi Layer Perceptron` amb Validació creuada

Mètrica	Valors
Accuracy	0.7262
Precision	0.5000
Recall	0.4928
F1-Score	0.4964

Taula 17: Resultats de classificació del model Multi Layer Perceptron

El MLP ha mostrat un comportament equilibrat entre precisió i exhaustivitat, amb un F1 semblant al KNN. Tanmateix, el seu rendiment no supera els millors models basats en arbres, com XGBoost o LightGBM. Aquest resultat pot indicar que el conjunt de dades no conté una complexitat no lineal tan elevada com per justificar completament l'ús d'una xarxa neuronal profunda, o bé que el model necessitaria més ajustos i/o dades per assolir el seu potencial complet.

El codi de l'entrenament es trobarà al punt 9.7.9 de l'annex.

6.9 Discussió de Resultats

Els resultats obtinguts per a cada model es mostren a la taula següent:

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.7984	0.6863	0.5000	0.5785
XGBoost	0.7905	0.6269	0.6000	**0.6131**
LightGBM	0.7984	0.6557	0.5714	0.6107
Logistic Regression	0.7143	0.4211	0.1159	0.1818
SVM	0.7579	0.6818	0.2174	0.3297
KNN	0.6984	0.4545	0.5072	0.4795
MLP (Xarxa neuronal)	0.7262	0.5000	0.4928	0.4964

Taula 18: Resultats de rendiment dels diferents models

Millors models: Tot i que diversos models han assolit una exactitud elevat (superior al 75%), aquest valor és poc informatiu donat el desequilibri del conjunt de dades. El criteri més rellevant en aquest cas és el **F1-Score**, ja que busca un bon equilibri entre precisió i sensibilitat.

El model que ha obtingut el millor F1-Score ha estat el **XGBoost**, amb un valor de **0.6131**. Aquest rendiment destaca per l'equilibri entre la seva precisió (0.6269) i sensibilitat (0.6000), mostrant una gran capacitat per detectar la classe minoritària amb un bon control dels falsos positius. Els models de boosting com XGBoost són coneguts per la seva potència en tasques de classificació binària, ja que combinen múltiples models febles per formar un model fort, aprofitant les seves fortaleses i corregint-ne les debilitats de forma iterativa.

Els models **LightGBM** i **Random Forest** també han mostrat un rendiment molt competitiu, amb F1-Scores de 0.6107 i 0.5785 respectivament. Tot i tenir un F1-Score lleugerament inferior, el Random Forest és un model robust i interpretable, especialment útil quan es treballa amb variables mixtes. La seva solidesa en contextos amb conjunts de dades desbalancejats, juntament amb l'ús de `class_weight='balanced'` i validació creuada, contribueixen a la seva eficàcia.

Models amb rendiment mitjà: El **MLP (Multi-Layer Perceptron)** ha aconseguit un F1-Score de 0.4964, similar al KNN. Aquest resultat, encara que no és dolent, pot estar limitat per la mida del conjunt de dades, ja que les xarxes neuronals solen necessitar més dades per mostrar el seu potencial. A més, tot i ajustar diversos hiperparàmetres, és possible que calgui més optimització o un pre-processament específic per millorar-ne el rendiment.

Models amb pitjor rendiment: El pitjor resultat pel que fa a F1-Score ha estat el de la **Regressió Logística**, amb un valor molt baix de **0.1818**. Aquest resultat reflecteix una gran dificultat del model per capturar les característiques que permeten detectar la classe minoritària, la qual cosa és habitual en problemes desbalancejats quan es fa servir un model lineal simple. Tot i tenir un *accuracy* relativament alt (0.7143), aquest valor és enganyós, ja que el model simplement tendeix a classificar gairebé tot com a classe majoritària.

De forma similar, el **SVM** ha obtingut una precisió elevada (0.6818), però amb una sensibilitat molt baixa (0.2174), cosa que es tradueix en un F1-Score de 0.3297. Aquest comportament pot indicar una classificació massa conservadora, penalitzant les prediccions positives (classe minoritària), possiblement degut a una configuració no òptima del paràmetre de marge o a la influència del desbalanceig.

A continuació es mostra una gràfica de la comparació del rendiment dels diferents models, per a tenir una referència visual més clara de les anteriors dades:

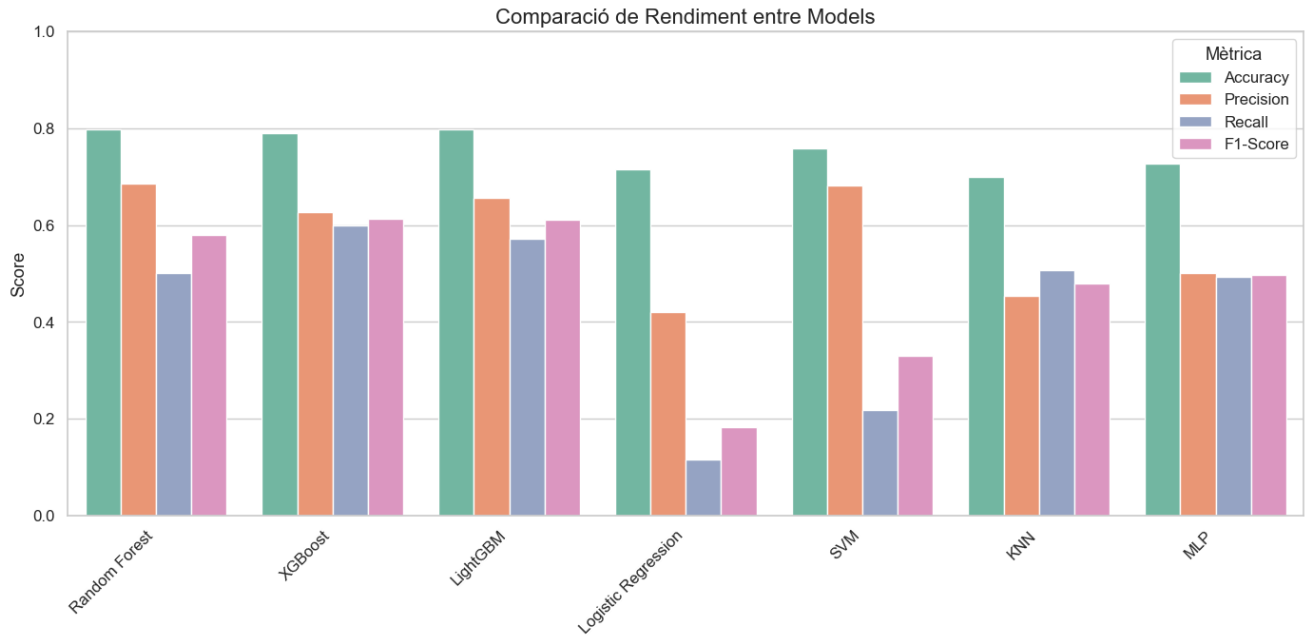

Figura 26: Gràfica comparativa del rendiment dels diferents models entrenats

En conjunt, els millors resultats han estat obtinguts amb models basats en arbres, especialment el **XGBoost**, que ha destacat amb el millor F1-Score, seguit molt de prop per LightGBM i Random Forest. Aquests models són especialment adequats per a problemes de classificació amb dades desbalancejades, com el present cas, i ofereixen un bon compromís entre rendiment i robustesa. Els models més senzills (Regressió Logística, KNN, SVM) han quedat per darrere, probablement per limitacions en la seva capacitat de capturar la complexitat del problema o per no gestionar bé el desbalanceig.
Tenint en compte l'objectiu de maximitzar el F1-Score i detectar millor la classe minoritària, la decisió més raonable seria escollir el **XGBoost**, que combina un rendiment elevat amb una bona capacitat de generalització.

7 Discussió i conclusions

7.1 Impacte de la Selecció de Característiques

La selecció i creació de característiques ha tingut un impacte decisiu en el rendiment dels models. Les característiques sintètiques agregades a nivell d'assaig han permès transformar canals de dades seqüencials en variables descriptives, facilitant l'ús de models de classificació estàndard. Tot i això, el rendiment obtingut suggereix que ha faltat un estudi més exhaustiu de les característiques per optimitzar la representació de la informació rellevant. És possible que característiques crítiques no hagin estat capturades adequadament, o que altres introduïssin soroll no desitjat.

A més, cal considerar que les etiquetes dels assaigs poden contenir cert grau de soroll, derivat de raons pràctiques com la manca de temps, decisions subjectives o criteris no homogenis entre enginyers. També pot passar que el comportament d'un prototip sigui erràtic en comparació a un vehicle homologat, però acceptable donada la situació, fent que alguns assaigs catalogats com a "bons" no ho siguin veritablement. Això complica la tasca dels models supervisats, ja que es poden veure forçats a aprendre patrons inconsistents.

Finalment, factors com el desgast del banc de proves, així com condicions ambientals o operacionals variables, poden introduir una variabilitat en les dades difícil de modelar o de detectar de manera supervisada. Aquest conjunt de limitacions ha pogut condicionar negativament la capacitat dels models per generalitzar.

7.2 Limitacions de l'Estudi

- **Qualitat i naturalesa de les dades:** El conjunt de dades presenta un desbalanceig entre classes i un nombre relativament reduït d'exemples, fet que dificulta la generalització dels models.
- **Sobreajustament:** Malgrat l'ús de tècniques com la validació creuada i la ponderació de classes, els models entrenats mostren signes de sobreajustament, fins i tot models com *Random Forest* el qual no tendeix a tenir-ne, possiblement relacionats amb un conjunt de dades limitat i una representació millorable de les característiques.
- **Subjectivitat en l'etiquetatge:** Les etiquetes poden contenir soroll degut a criteris no homogenis o decisions apreses en condicions pràctiques que no sempre reflecteixen fidelment la qualitat real dels assaigs.
- **Condicions externes no controlades:** Variacions operacionals, desgast dels equips o canvis ambientals poden afectar la forma de les gràfiques i influir en el comportament del sistema de classificació sense ser fàcilment detectables o corregibles.

7.3 Implicacions dels Resultats

Els resultats suggereixen que és tècnicament viable plantejar un sistema de classificació automàtica basat en dades experimentals, però la seva aplicabilitat real dependrà fortament de la qualitat, consistència i quantitat de dades disponibles. Caldria també disposar d'una definició més clara i robusta de les etiquetes per evitar inconsistències en el procés d'entrenament. Amb les condicions actuals, els models supervisats ofereixen millor rendiment que els no supervisats, però amb riscos significatius de sobreajustament.

7.4 Conclusió

Aquest estudi ha tingut com a objectiu principal determinar si, a partir de dades extretes de gràfiques d'assaigs experimentals, és possible desenvolupar un sistema de classificació robust i eficient.

S'han aplicat tècniques de selecció i construcció de característiques, entrenant tant models supervisats com no supervisats. El model Random Forest, tot i mostrar rendiment competitiu, ha evidenciat certes limitacions, igual que altres models supervisats com XGBoost o LightGBM. En general, els models supervisats han superat els no supervisats, però també han mostrat una sensibilitat elevada al sobreajustament, fet que compromet la seva robustesa.

Aquest fet pot estar relacionat amb la manca d'un estudi més profund de les característiques, la presència de soroll en les etiquetes, o condicions externes que introdueixen variabilitat difícil de controlar. El comportament erràtic però acceptable de certs prototips i les condicions operacionals del banc de proves també han pogut distorsionar les dades.

En aquest sentit, els resultats apunten que sí, és potencialment viable construir un sistema de classificació eficaç, però les condicions actuals no permeten garantir-ne la robustesa. Així, cal considerar aquest treball com una base per a estudis més complets que abordin aquestes limitacions.

7.5 Línies Futures

- **Ampliació del conjunt de dades:** Recollir més assaigs per millorar la representativitat i facilitar l'aprenentatge de patrons generals.
- **Millora de les característiques:** Estudiar en profunditat la rellevància i la capacitat discriminativa de cada característica, i explorar tècniques avançades d'extracció automàtica.
- **Data augmentation:** Aplicar tècniques de generació sintètica de dades per enriquir el conjunt d'entrenament i millorar la generalització.
- **Redefinició i validació de les etiquetes:** Establir criteris més clars i consistents per a l'etiquetatge, i considerar mecanismes de consens entre tècnics o anotació múltiple.
- **Combinació de models:** Explorar enfocaments híbrids que combinin models supervisats i no supervisats, o sistemes d'ensamblatge per aprofitar les fortaleses de cada tipus.
- **Anàlisi qualitativa:** Incorporar eines de visualització i anàlisi d'errors que ajudin a entendre millor els casos problemàtics.

Aquestes línies futures poden ajudar a transformar un prototip prometedor en una solució realment fiable per a la classificació automàtica en entorns d'assaigs experimentals.

8 Eina complementària: *Error Log Tool*

Després d'haver dut a terme un estudi comparatiu entre diferents models de classificació per detectar assaigs defectuosos, es va constatar que, tot i que enfocaments supervisats com `Random Forest` o `XGBoost` oferien resultats prometedors, el seu rendiment no era suficient per a una l'aplicació pràctica. En concret, es van identificar tres limitacions principals que dificultaven l'entrenament de models supervisats robustos:

- Pocs arxius de dades etiquetats.
- Etiquetatge a nivell d'assaig, pel que no es dona informació al model quin canal de dades, o canals, han estat els responsables de que l'assaig hagi sortit malament.
- Existència d'assaigs on l'etiquetatge no es del tot fiable, ja que, en ocasions, les decisions es prenen de forma ràpida i poc sistematitzada, introduint soroll en les dades.
- Alta variabilitat en els resultats dels assaigs, deguda a la naturalesa dels prototips o vehicles que el seu comportament s'allunya de la mitjana, el desgast del banc de proves i altres factors ambientals o operatius que escapen al control dels models supervisats.

Davant d'aquestes dificultats, i com a part de la col·laboració directa amb el subdepartament implicat, es va plantejar una alternativa enfocada a la millora del procés d'etiquetatge. Aquesta alternativa consisteix en el desenvolupament d'una **eina interactiva amb interfície gràfica** dissenyada per facilitar als enginyers la identificació manual de les gràfiques errònies dins de cada assaig experimental.
Aquesta eina permet, de forma intuïtiva, que l'usuari marqui quines gràfiques considera incorrectes per a cada arxiu `.tab`, i genera automàticament un arxiu `.csv` estructurat amb els noms dels assaigs i les gràfiques corresponents que han estat etiquetades com errònies durant un projecte (conjunt d'assaigs). Amb aquest procediment es busca iniciar la construcció d'un conjunt de dades supervisades de més qualitat per d'aquesta manera, millorar la traçabilitat dels errors dins dels assaigs i reduir el soroll en les etiquetes.
Aquesta iniciativa representa un primer pas cap a una millor estructuració del procés d'etiquetatge i compleix amb la visió plantejada de l'apartat de línies futures del treball, obrint la porta a estudis més precisos i a l'entrenament de models més robusts en un futur.

8.1 Descripció del Funcionament del Codi

L'eina desenvolupada, anomenada `ErrorLog Tool`, ha estat implementada en Python i combina l'ús de diverses llibreries per oferir una interfície interactiva, amb un disseny semblant a un *software* amb el que els enginyers del subdepartament validen les gràfiques dels assaigs, per així facilitar la integració de la nova eina.

Llibreries clau:

- `Tkinter` [12]: per a la construcció de la interfície gràfica d'usuari.
- `matplotlib`: per visualitzar gràfiques amb possibilitat de selecció interactiva.
- `watchdog`: per monitoritzar l'arribada de nous fitxers `.tab` en un directori.
- `pandas`, `numpy`, `re`: per manipulació i extracció de dades dels arxius.

El codi està estructurat en tres blocs principals:

1. **Lectura i preprocessat de fitxers**: es llegeixen fitxers `.tab`, obtenint tots els canals de dades de l'arxiu.

2. **Creació de la interfície gràfica**: mostrant les gràfiques agrupades per categories (Desplaçaments, forces, moments i dades específiques de cada assaig), on l'usuari pot seleccionar visualment quines presenten errors.

3. **Generació i actualització d'un fitxer CSV**: Recull per cada assaig les gràfiques marcades com errònies.

8.2 Funcionament de l'Eina

Un cop iniciada l'eina, mitjançant una primera interfície es demana a l'usuari:

- Seleccionar un directori de monitorització (on s'aniran afegint arxius `.tab`).
- Seleccionar o crear un fitxer CSV on quedarà el registre de les gràfiques errònies detectades per cada assaig del projecte.

A continuació es mostra una il·lustració de la primera interfície del programa:

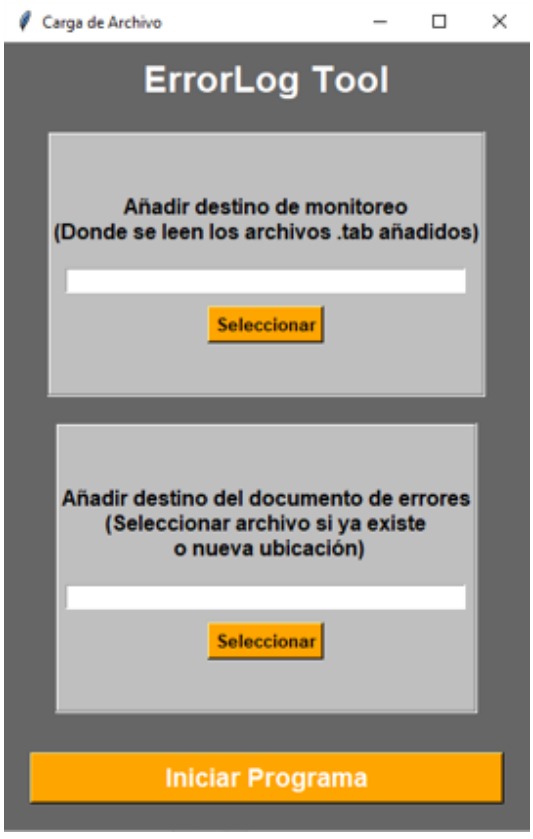

Figura 27: Primera interífcie del programa

Un cop afegida la informació pertinent, es tancarà la primera interfície i s'obrirà una altre. En aquesta segona interfície el primer que s'observa és una pantalla totalment en gris la qual conté a la part superior esquerre un botó de color vermell amb el text: *Finalizar*, aquest botó serveix per sortir del programa, a la part superior central hi apareix el text: *Nombre del Ensayo*, aquest títol s'anirà actualitzant a mesura que detecti arxius de dades nous i finalment a la part superior dreta, un botó verd amb el text *Send (Enviar)*, la funció d'aquest últim botó és enviar totes les gràfiques que en aquell moment estiguin seleccionades cap a l'arxiu .csv.
Aquesta segona interfície apareix totalment grisa degut a que en un inici no té cap arxiu de dades a llegir, i

en l'instant que en detecta un la interfície s'actualitz mostrant les gràfiques pertinents de l'assaig detectat.

A continuació es mostra la segona interfície amb un exemple d'un arxiu de dades real.

Aquest contingut ha estat retirat per confidencialitat

Figura 28: Segona interfície del programa

En la imatge s'observen dues gràfiques del mateix estil, però una ha estat marcada en vermell degut a que s'ha detectat soroll no desitjat. Un cop s'analitzin totes les gràfiques de l'assaig, es donarà al botó de *Send*, escrivint així el nom de l'assaig juntament amb les gràfiques detectades com a dolentes. Si no es detecta res anòmal, simplement no cal fer res, ja que al afegir el següent arxiu de dades del pròxim assaig la interfície s'actualitza sola.
L'arxiu .csv pren el següent format:

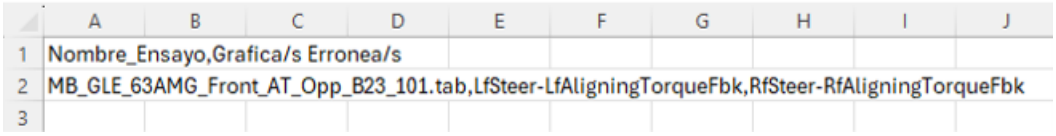

Figura 29: Format de l'arxiu csv generat

En primer lloc apareix el nom de l'assaig seguit per les gràfiques seleccionades, les gràfiques queden escrites de la següent manera: `eix_Y-eix_X`, ja que hi ha gràfiques que poden compartir l'eix y. Com el nom de *csv* indica: (*coma separated values*), les variables s'escriuen separades per comes. No es el format més atractiu visualment, però si un dels més útils si en un futur es decideix implementar una eina per llegir les dades.

En resum, aquesta eina permet capturar coneixement expert de manera semi-automatitzada, millorant la qualitat de les etiquetes, passant de tenir un etiquetatge a nivell d'assaig global a tenir un etiquetatge a nivell gràfica, i facilitant la creació d'un conjunt d'entrenament realment representatiu.

9 Annexos

9.1 Codi de recollida i neteja de dades

```
Aquest contingut ha estat retirat per confidencialitat
```

9.2 Codi de lectura de les dades

```
Aquest contingut ha estat retirat per confidencialitat
```

9.3 Identificació de Projectes Classificats

```
Aquest contingut ha estat retirat per confidencialitat
```

9.4 Separació de dades Bones i Dolentes

```
Aquest contingut ha estat retirat per confidencialitat
```

9.5 Classificació per tipus d'Assaig

```
Aquest contingut ha estat retirat per confidencialitat
```

9.6 Codi de l'estructura de les dades d'entrenament

9.6.1 Isolation Forest

Features

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import numpy as np

# Extracció de característiques basades en pics d'una senyal normalitzada
def peak_det(data, current_data):
    # Normalitzem la senyal per tenir una escala comuna
    current_data =
    ↪  StandardScaler().fit_transform(current_data.to_numpy().reshape(-1,
    ↪  1)).flatten()

    # Derivada per capturar canvis sobtats
    diff = np.diff(current_data)

    # Càlcul de la relació entre el pic màxim i la desviació estàndard
    std_diff = diff.std()
    max_diff = max(diff.max(), abs(diff.min()))
    pd1 = max_diff / std_diff

    return {'pd': pd1}


# Extracció de característiques de senyals que haurien de romandre a prop de
↪  zero
def zero_signal_features(data, current_data):
    # Definim llindars segons el tipus de senyal
    if data in ['LfRollPositionFbk', 'RfRollPositionFbk']:
        high, low = 0.01, 0.005
    elif data in ['SteeringWheelPositionFbk', 'SteeringWheelAngle']:
        high, low = 0.1, 0.05
    elif data == 'SteeringWheelForceFbk':
        high, low = 0.5, 0.2
    elif data in ['LfLatForceFbk', 'RfLatForceFbk', 'LfLongForceFbk',
    ↪  'RfLongForceFbk']:
        high, low = 100, 15
    else:
        high, low = 8, 2  # AligningTorque

    # Reajustem les senyals amb mínim diferent de zero
    if data in ['SteeringWheelForceFbk', 'SteeringWheelAngle',
    ↪  'SteeringWheelPositionFbk', 'SteeringWheelTorque']:
        current_data = current_data - current_data.min()

    # Variables sintètiques simples: màxim, mínim i energia (àrea sota la corba
    ↪  al quadrat)
```

```
    area = np.sum(abs(current_data)**2)
    return {'min': current_data.min(), 'max': current_data.max(), 'area': area}


# Correlació entre cicles de senyal
def cycle_correl(data, current_data):
    # Normalitzem la senyal
    current_data =
    ↪  StandardScaler().fit_transform(current_data.to_numpy().reshape(-1,
    ↪  1)).flatten()

    mid = len(current_data) // 2
    quart = len(current_data) // 4

    ciclo1 = current_data[:mid]
    ciclo2 = current_data[mid:mid + len(ciclo1)]

    upper_cycle = current_data[:quart]
    lower_cycle = current_data[quart:2 * quart]

    # Correlació entre dues meitats i diferència entre cicles
    corr_ciclos = np.corrcoef(ciclo1, ciclo2)[0, 1]
    cycle_sum = np.sum(upper_cycle) - np.sum(lower_cycle)

    return {'corr_ciclos': corr_ciclos, 'Diferencia_anada_tornada': cycle_sum}


# Característiques específiques per forces verticals
def vert_force_feature(data, current_data):
    # Normalització
    current_data =
    ↪  StandardScaler().fit_transform(current_data.to_numpy().reshape(-1,
    ↪  1)).flatten()

    # Derivada de la senyal i valors absoluts per detectar variabilitat
    diff = np.diff(current_data)
    diff = np.concatenate([diff, [0]])  # evitem perdre un valor per longitud
    abs_diff = abs(diff)

    # Mitjana i desviació estàndard de la variació
    mean = abs_diff.mean()
    stds = abs_diff.std()

    return {'abs_diff_vert_mean': mean, 'abs_diff_vert_std': stds}
```

Generar estructura

```
    # --- Imports necessaris ---
import os
import numpy as np
import pandas as pd

# Defineixen els camins de les carpetes amb dades verticals (assiags bons,
↪  dolents i sense classificar)
verts_paths = [
    'C:/Users/AT018393/Desktop/Tabs/Vert',
    'C:/Users/-/Desktop/Tabs/Vert/No Vert',
    'C:/Users/-/Desktop/Incompletos/Vert'
]

all_data = []  # Llista on s'acumularan totes les dades sintètiques
index = 0  # Comptador per a assignar ID únic a cada gràfica

# Iterem per cada carpeta d'assaigs verticals
for vert_path in verts_paths:
    # Assignem la classe (0 = vert correcte, 1 = no vert)
    class_info = {"class": 1} if 'No Vert' in vert_path else {"class": 0}

    # Llistem els fitxers dins la carpeta
    for vert in os.listdir(vert_path):
        if 'no vert' in vert.lower():
            continue  # Ometem arxius que indiquen explícitament "no vert"

        ensayo_path = os.path.join(vert_path, vert)

        try:
            df_after_Y = read_tab(ensayo_path)  # Llegim el fitxer de dades
        except:
            print(f"No es pot llegir: {vert}")
            continue

        ensayo_info = {'ensayo_id': vert}  # Identificador de l'assaig
        index = 0  # Reiniciem ID de gràfica per a cada fitxer

        # --- Tractament de canals normals (ex. posicions de roda) ---
        for data in normal_data:
            grafica_info = {'graph': data, 'graph_id': index}
            index += 1

            current_data = df_after_Y.loc[5:, data].astype(float)  # Extracció de
            ↪  la senyal

            try:
                features = peak_det(data, current_data)  # Extracció de
                ↪  característiques
```

```
            except:
                print(f"No vàlid: {vert}")
                break

            # Ajuntem tota la informació i afegim a la llista final
            all_data.append({**ensayo_info, **grafica_info, **features,
            ↪  **class_info})

        # --- Tractament de senyals que poden tenir senyal zero ---
        canales = df_after_Y.head(0)

        if 'SteeringWheelPositionFbk' in canales:
            zero_data = [
                'LfRollPositionFbk', 'RfRollPositionFbk',
                ↪  'SteeringWheelPositionFbk',
                'SteeringWheelForceFbk', 'LfLatForceFbk', 'RfLatForceFbk',
                'LfLongForceFbk', 'RfLongForceFbk', 'LfAligningTorqueFbk',
                ↪  'RfAligningTorqueFbk'
            ]
        else:
            zero_data = [
                'LfRollPositionFbk', 'RfRollPositionFbk', 'SteeringWheelAngle',
                'SteeringWheelTorque', 'LfLatForceFbk', 'RfLatForceFbk',
                'LfLongForceFbk', 'RfLongForceFbk', 'LfAligningTorqueFbk',
                ↪  'RfAligningTorqueFbk'
            ]

        for data in zero_data:
            # Mapeig de noms antics a nous per coherència
            if data == 'SteeringWheelAngle':
                grafica_info = {'graph': 'SteeringWheelPositionFbk', 'graph_id':
                ↪  index}
            elif data == 'SteeringWheelTorque':
                grafica_info = {'graph': 'SteeringWheelForceFbk', 'graph_id':
                ↪  index}
            else:
                grafica_info = {'graph': data, 'graph_id': index}
            index += 1

            current_data = df_after_Y.loc[5:, data].astype(float)

            features = zero_signal_features(data, current_data)
            all_data.append({**ensayo_info, **grafica_info, **features,
            ↪  **class_info})

        # --- Tractament de senyals verticals (forces, toe, camber, etc.) ---
        vert_data = ['LfVertForceFbk', 'RfVertForceFbk', 'LfToe', 'RfToe',
        ↪  'LfCamber', 'RfCamber', 'LfY0', 'RfY0']
        vert_xdata = ['LfZ0', 'RfZ0']
```

```
        for data in vert_data:
            grafica_info = {'graph': data, 'graph_id': index}
            index += 1

            x_axis = 'LfZ0' if 'Lf' in data else 'RfZ0'

            current_data = df_after_Y.loc[5:, data].astype(float)
            x_data = df_after_Y.loc[5:, x_axis].astype(float)

            try:
                features = peak_det(data, current_data)
                features_2 = cycle_correl(data, current_data)
            except:
                print(f"No vàlid: {vert}")
                break

            if 'VertForce' in data:
                try:
                    features_3 = vert_force_feature(data, current_data)
                    features = {**features, **features_2, **features_3}
                except:
                    print(f"No vàlid: {vert}")
                    break
            else:
                features = {**features, **features_2}

            all_data.append({**ensayo_info, **grafica_info, **features,
            ↪  **class_info})

# Convertim tota la informació a DataFrame final
df = pd.DataFrame(all_data)


# --- Exportació a Excel: cada gràfica en una pestanya separada ---
def excel_indiv_vert_features(df, output_name):
    sub_dfs = {}

    # Agrupem per ID de gràfica
    for gid in df['graph_id'].unique():
        sub_df = df[df['graph_id'] == gid].copy()
        sub_df = sub_df.dropna(axis=1, how='all')  # Eliminem columnes
        ↪  completament buides
        sub_dfs[gid] = sub_df

    # Guardem a Excel
    with pd.ExcelWriter(output_name, engine='xlsxwriter') as writer:
        for gid, sdf in sub_dfs.items():
            sheet_name = f"graph_{gid}"[:31]  # Nom màxim permès per Excel
```

```
            sdf.to_excel(writer, sheet_name=sheet_name, index=False)

    print(f"Fitxer guardat com: {output_name}")

# Cridem a la funció per exportar les dades
excel_indiv_vert_features(df, 'IsolationForest3.xlsx')
```

9.6.2 Random Forest

Features

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import numpy as np

# Feature dels conjunts de dades que segueixen un patró clar
def peak_det(data, current_data, index):
    # Normalització z-score per establir un criteri comú
    stdscaler = StandardScaler()
    current_data = stdscaler.fit_transform(current_data.to_numpy().reshape(-1,
    ↪  1)).flatten()

    # Derivada i càlcul del màxim canvi relatiu
    diff = np.diff(current_data)
    max_diff = max(diff.max(), abs(diff.min()))
    pd1 = max_diff / std_diff

    feature = {f'pd_{index}': pd1}
    return feature

# Variables de les dades que han de ser properes a 0
def zero_signal_features(data, current_data, index):
    # Ajustar mínim per certes variables específiques (com la força o angle de
    ↪  direcció)
    if data in ['SteeringWheelForceFbk', 'SteeringWheelAngle',
                'SteeringWheelPositionFbk', 'SteeringWheelTorque']:
        current_data = current_data - current_data.min()

    max_val = current_data.max()
    min_val = current_data.min()
    area = np.sum(abs(current_data)**2)

    features = {
        f'max_{index}': max_val,
        f'min_{index}': min_val,
        f'area_{index}': area
    }
    return features

# Variables de es dades específiques de l'assaig vertical
def cycle_correl(data, current_data, index):
    stdscaler = StandardScaler()
    current_data = stdscaler.fit_transform(current_data.to_numpy().reshape(-1,
    ↪  1)).flatten()

    # Dividim la senyal en dues meitats per comparar cicles
    mid = len(current_data) // 2
    quart = len(current_data) // 4
```

```
    ciclo1 = current_data[:mid]
    ciclo2 = current_data[mid:mid + len(ciclo1)]

    upper_cycle = current_data[:quart]
    lower_cycle = current_data[quart:2 * quart]

    # Correlació entre dues meitats (anada i tornada)
    corr_ciclos = np.corrcoef(ciclo1, ciclo2)[0, 1]
    cycle_sum = np.sum(upper_cycle) - np.sum(lower_cycle)

    features = {
        f'corr_ciclos_{index}': corr_ciclos,
        f'Diferencia_anada_tornada_{index}': cycle_sum
    }
    return features

# Variables de es dades específiques de l'assaig vertical
def vert_force_feature(data, current_data, index):
    stdscaler = StandardScaler()
    current_data = stdscaler.fit_transform(current_data.to_numpy().reshape(-1,
    ↪  1)).flatten()

    # Derivada absoluta i mètriques associades
    diff = np.diff(current_data)
    diff = np.concatenate([diff, [0]])  # S'afegeix 0 per mantenir la longitud
    ↪  original

    abs_diff = abs(diff)
    mean = abs_diff.mean()
    stds = np.std(abs_diff)

    features = {
        f'abs_diff_vert_mean_{index}': mean,
        f'abs_diff_vert_std_{index}': stds
    }
    return features
```

Generar estructura

```
    import os
import numpy as np
import pandas as pd

# Rutes als assajos bons i dolents
no_vert = 'C:/Users/-/Desktop/Tabs/Vert/No Vert'
vert = 'C:/Users/-/Desktop/Tabs/Vert'

all_paths = [vert, no_vert]

all_data = []  # Aquí guardarem totes les files amb les dades finals
ind = 0  # Index per a identificar els canals

# Iterem per cada carpeta (categoria)
for path in all_paths:

    # Assignem l'etiqueta de classificació segons si és un assaig "No Vert"
    class_info = {"class": 1} if 'No Vert' in path else {"class": 0}

    # Llistem tots els subcarpetes (assajos)
    assajos = os.listdir(path)

    for nom_assaig in assajos:
        # Excloem carpetes amb 'no vert' dins del nom per seguretat
        if 'no vert' in nom_assaig.lower():
            continue

        # Llegim el fitxer tabulat
        ruta_assaig = os.path.join(path, nom_assaig)
        df = read_tab(ruta_assaig)
        index = 0

        # Identificador de l'assaig
        info_assaig = {'ensayo_id': nom_assaig}
        features_assaig = {}

        # Senyals amb patróns clars
        for canal in normal_data:
            index += 1
            current_data = df.loc[5:, canal].astype(float)

            try:
                feats = peak_det(canal, current_data, index)
            except:
                print('Error al canal (pic):', nom_assaig)
                break

            features_assaig.update(feats)
```

```
        # Canals amb informació propera al zero
        canals = df.head(0)
        #En arxius de dades antics hi ha noms de canals que canvien per això es
        ↪  fa aquesta comprovació
        if 'SteeringWheelPositionFbk' in canals:
            zero_channels = ['LfRollPositionFbk', 'RfRollPositionFbk',
            ↪  'SteeringWheelPositionFbk', 'SteeringWheelForceFbk',
                             'LfLatForceFbk', 'RfLatForceFbk', 'LfLongForceFbk',
                             ↪  'RfLongForceFbk',
                             'LfAligningTorqueFbk', 'RfAligningTorqueFbk']
        else:
            zero_channels = ['LfRollPositionFbk', 'RfRollPositionFbk',
            ↪  'SteeringWheelAngle', 'SteeringWheelTorque',
                             'LfLatForceFbk', 'RfLatForceFbk', 'LfLongForceFbk',
                             ↪  'RfLongForceFbk',
                             'LfAligningTorqueFbk', 'RfAligningTorqueFbk']

        for canal in zero_channels:
            index += 1
            current_data = df.loc[5:, canal].astype(float)

            feats = zero_signal_features(canal, current_data, index)
            features_assaig.update(feats)

        # Canals de dades assaig vertical
        vert_canals = ['LfVertForceFbk', 'RfVertForceFbk', 'LfToe', 'RfToe',
        ↪  'LfCamber', 'RfCamber', 'LfY0', 'RfY0']
        z_ref = {'Lf': 'LfZ0', 'Rf': 'RfZ0'}

        for canal in vert_canals:
            index += 1
            current_data = df.loc[5:, canal].astype(float)
            x_canal = z_ref['Lf'] if 'Lf' in canal else z_ref['Rf']
            x_data = df.loc[5:, x_canal].astype(float)

            try:
                feats1 = peak_det(canal, current_data, index)
                feats2 = cycle_correl(canal, current_data, index)
            except:
                print('Error al canal vertical:', nom_assaig)
                break

            # Si és força vertical, afegim també la variabilitat de derivades
            if 'VertForce' in canal:
                try:
                    feats3 = vert_force_feature(canal, current_data, index)
                    feats_comb = {**feats1, **feats2, **feats3}
                except:
```

```
                    print('Error derivada vertical:', nom_assaig)
                    break
            else:
                feats_comb = {**feats1, **feats2}

            features_assaig.update(feats_comb)

        # Compilar tota la informació de l'assaig en un diccionari únic
        registre_final = {**info_assaig, **features_assaig, **class_info}
        all_data.append(registre_final)

# Un cop acabada la lectura de tots els assajos, convertim a DataFrame
df_final = pd.DataFrame(all_data)

# Guardem el conjunt de dades a Excel
df_final.to_excel('RandomForestTrain.xlsx', index=False)
```

9.7 Codi d'entrenament i resultats dels Models

9.7.1 Isolation Forest - Entrenament

```
import pandas as pd
import numpy as np
from sklearn.ensemble import IsolationForest
import matplotlib.pyplot as plt

# Llegim totes les fulles de l'arxiu Excel
path_excel = 'IsolationForestTrain.xlsx'
all_sheets = pd.read_excel(path_excel, sheet_name=None)
df_list = list(all_sheets.values())       # Llista de tots els dataframes per
↪  fulla
sheet_names = list(all_sheets.keys())     # Noms de les fulles

# Columnes que no són característiques (features)
no_features = ['ensayo_id', 'graph', 'graph_id', 'class', 'Unnamed: 0']

# Aplicació del model Isolation Forest a cada tipus de gràfica
resultats_anomalies = []

for idx, df in enumerate(df_list):
    nom_grafica = sheet_names[idx]

    # Preprocessament: selecció de columnes vàlides i conversió de tipus
    feature_cols = df.columns.difference(no_features)
    df[feature_cols] = df[feature_cols].apply(pd.to_numeric, errors='coerce')  #
    ↪  Convertim a numèric, errors a NaN
    df = df.fillna(-999)  # Substituïm NaNs per -999 per al model

    X = df[feature_cols]

    # Entrenament del model Isolation Forest
    iso_model = IsolationForest(contamination=0.05, random_state=42)  # 5% de les
    ↪  dades considerades outliers
    iso_model.fit(X)
    df['anomaly_score'] = iso_model.decision_function(X)  # Com més baix, més
    ↪  anòmal
    df['is_anomaly'] = iso_model.predict(X)  # -1 = anomalia, 1 = normal

    # Convertim a una "probabilitat" inversa per facilitar la interpretació
    df['anomaly_prob'] = -df['anomaly_score']

    # Guardem resultats
    df['grafica'] = nom_grafica
    resultats_anomalies.append(df[['ensayo_id', 'graph', 'graph_id',
    ↪  'anomaly_prob', 'is_anomaly'] + list(feature_cols)])
```

```
# Separació entre dades normals i anòmales (opcional per a visualització)
normal = df[df['is_anomaly'] == 1]
anomalies = df[df['is_anomaly'] == -1]

# Les següents seccions comentades són útils per visualitzar els resultats
↪  segons certes característiques.
# Pots descomentar-les si vols veure gràfics específics de les anomalies.

# if 'pd' in df.columns:
#     plt.figure(figsize=(20, 5))
#     plt.plot(normal['pd'], 'o', label='Normal')
#     plt.plot(anomalies['pd'], 'o', label='Anomalia')
#     plt.ylabel("pd")
#     plt.title(df['graph'][1])
#     plt.legend()
#     plt.show()

# elif 'area' in df.columns:
#     plt.figure(figsize=(20, 5))
#     plt.plot(normal['area'], 'o', label='Normal')
#     plt.plot(anomalies['area'], 'o', label='Anomalia')
#     plt.ylabel("area")
#     plt.yscale('log')
#     plt.title(df['graph'][1])
#     plt.legend()
#     plt.show()

#     # Comparació entre mínims i màxims
#     plt.figure(figsize=(20, 5))
#     plt.scatter(normal['max'], normal['min'], label='Normal')
#     plt.scatter(anomalies['max'], anomalies['min'], label='Anomalia')
#     plt.xlabel("max")
#     plt.ylabel("min")
#     plt.title(df['graph'][1])
#     plt.legend()
#     plt.show()


# if 'Diferencia_anada_tornada' in df.columns:
#     plt.figure(figsize=(20, 5))
#     plt.scatter(normal['corr_ciclos'], normal['Diferencia_anada_tornada'],
↪  label='Normal')
#     plt.scatter(anomalies['corr_ciclos'],
↪  anomalies['Diferencia_anada_tornada'], label='Anomalia')
#     plt.xlabel("anada tornada")
#     plt.ylabel("ciclos")
#     plt.title(df['graph'][1])
#     plt.legend()
#     plt.show()
```

```
    #     if 'abs_diff_vert_mean' in df.columns:
    #         plt.figure(figsize=(20, 5))
    #         plt.scatter(normal['abs_diff_vert_mean'],
    ↪  normal['abs_diff_vert_std'], label='Normal')
    #         plt.scatter(anomalies['abs_diff_vert_mean'],
    ↪  anomalies['abs_diff_vert_std'], label='Anomalia')
    #         plt.xlabel("std")
    #         plt.ylabel("mean")
    #         plt.title(df['graph'][1])
    #         plt.legend()
    #         plt.show()

# Unim tots els resultats en un únic DataFrame
df_anomalias_total = pd.concat(resultats_anomalies, ignore_index=True)

# Ordenem pel nivell d'anomalia (de més anòmal a menys)
df_anomalias_total = df_anomalias_total.sort_values(by='anomaly_prob',
↪  ascending=False)

# Mostrem les gràfiques més anòmales (opcional)
# print(df_anomalias_total.head(10))
```

9.7.2 Isolation Forest - Resultats

```
    from sklearn.metrics import confusion_matrix, precision_score, recall_score,
    ↪  f1_score, accuracy_score
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os

# Filtrar només les gràfiques classificades com a anòmales
anomalias = df_anomalias_total[df_anomalias_total['is_anomaly'] == -1]

# Agrupar per assaig i recollir les dades rellevants de les gràfiques anòmales
resultat_ensayos = anomalias.groupby('ensayo_id').agg({
    'graph': list,
    'anomaly_prob': list,
    'pd': list,
    'area': list,
    'min': list,
    'max': list,
    'corr_ciclos': list,
    'Diferencia_anada_tornada': list,
    'abs_diff_vert_mean': list,
    'abs_diff_vert_std': list
}).reset_index()

# Obtenir la llista d'assaigs amb almenys una gràfica anòmala
ensayos_anomalos = resultat_ensayos['ensayo_id'].tolist()

print("Assaigs amb almenys una gràfica anòmala:")
print(resultat_ensayos)
print(ensayos_anomalos)

# Carregar llista d'assaigs amb dades incompletes des d'un directori
extra = 'C:/Users/AT018393/Desktop/Incompletos/Vert'
incompletos = os.listdir(extra)

# Eliminar els assaigs incomplets de la llista d'assaigs anòmals
ensayos_anomalos_etiquetados = [e for e in ensayos_anomalos if e not in
↪  incompletos]

# Mostrar estadístiques
print("Total d'anòmals:", len(ensayos_anomalos))
print("Anòmals amb dades completes:", len(ensayos_anomalos_etiquetados))
print("Assaigs eliminats per dades incompletes:", len(ensayos_anomalos) -
↪  len(ensayos_anomalos_etiquetados))

# Filtrar el dataframe per eliminar els assaigs incomplets
```

```
df_anom = df.copy()
df_anom = df_anom[~df_anom['ensayo_id'].isin(incompletos)]

# Obtenir classe real per cada assaig únic
ensayo_classes = df_anom[['ensayo_id', 'class']].drop_duplicates()

# Crear vectors de veritat real (y_true) i predicció (y_pred)
y_true = []
y_pred = []

for ensayo_id in ensayo_classes['ensayo_id']:
    # Classe real
    true_class = ensayo_classes.loc[ensayo_classes['ensayo_id'] == ensayo_id,
    ↪  'class'].values[0]
    y_true.append(true_class)

    # Predicció: 1 si considerat anòmal, 0 si no
    pred_class = 1 if ensayo_id in ensayos_anomalos_etiquetados else 0
    y_pred.append(pred_class)

# Calcular matriu de confusió i mètriques d'avaluació
conf_mat = confusion_matrix(y_true, y_pred)
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)
f1 = f1_score(y_true, y_pred)
accuracy = accuracy_score(y_true, y_pred)

# Mostrar resultats per consola
print("Matriu de Confusió:")
print(conf_mat)
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1:.4f}")

# Visualització de la matriu de confusió
sns.heatmap(conf_mat, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Bo', 'Dolent'], yticklabels=['Bo', 'Dolent'])
plt.xlabel('Predicció')
plt.ylabel('Classe real')
plt.title('Matriu de Confusió')
plt.show()
```

9.7.3 Random Forest - Entrenament i Resultats

```
    # Importació de llibreries necessàries
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import (
    classification_report, confusion_matrix, roc_curve, auc,
    precision_score, recall_score, f1_score, accuracy_score
)
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Carregar el fitxer Excel amb les dades del conjunt d'entrenament i test
path_excel = 'RandomForestTrain.xlsx'
df = pd.read_excel(path_excel)

# Definir quines columnes NO són variables predictives
no_features = ['ensayo_id', 'class', 'Unnamed: 0']

# Obtenir la llista de columnes que sí són variables predictives
feature_cols = df.columns.difference(no_features)

# Assegurar que totes les variables predictives siguin numèriques
df[feature_cols] = df[feature_cols].apply(pd.to_numeric, errors='coerce')

# Assignar les dades (X) i etiquetes (y)
X = df[feature_cols]
y = df['class']

# Eliminar files amb valors nuls i mantenir la consistència entre X i y
X = X.dropna()
y = y[X.index]

# Dividir el dataset en entrenament i test (80% / 20%), mantenint l'equilibri de
↪  classes
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y)

# Crear el model Random Forest amb hiperparàmetres definits
model = RandomForestClassifier(
    n_estimators=200,            # Nombre d'arbres del bosc aleatori
    random_state=42,             # Llavor per assegurar reproducció
    class_weight='balanced',     # Ajustar pes de les classes per compensar
    ↪  desbalanceig
    n_jobs=-1                    # Utilitzar tots els nuclis disponibles per
    ↪  entrenar en paral·lel
)
```

```
# Entrenament del model amb les dades d'entrenament
model.fit(X_train, y_train)

# Fer prediccions sobre el conjunt de test
y_pred = model.predict(X_test)

# Obtenir probabilitats de predicció per la classe positiva (dolent)
y_pred_proba = model.predict_proba(X_test)[:, 1]

# Imprimir l'informe de classificació detallat
print("\nClassification Report:")
print(classification_report(y_test, y_pred, digits=4))

# Calcular mètriques d'avaluació
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)

# Mostrar resultats per consola
print(f"\n Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")
print(f"Accuracy: {accuracy:.4f}")

# Crear i mostrar la matriu de confusió
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=["Bo", "Dolent"],
            yticklabels=["Bo", "Dolent"])
plt.title("Matriu de Confusió")
plt.xlabel("Predicció")
plt.ylabel("Real")
plt.show()

# Generar la corba ROC i calcular l'àrea sota la corba (AUC)
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
roc_auc = auc(fpr, tpr)

# Mostrar la corba ROC
plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC =
↪  {roc_auc:.4f})')
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')  # Línia diagonal
↪  (classificador aleatori)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
```

```
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.grid()
plt.show()

# Imprimir el valor de l'AUC
print(f"\nROC AUC: {roc_auc:.4f}")

# Calcular la importància de cada variable predictiva segons el model entrenat
importancias = model.feature_importances_
features = X.columns

# Crear DataFrame amb les importàncies i ordenar-les
df_importancia = pd.DataFrame({
    'feature': features,
    'importancia': importancias
}).sort_values(by='importancia', ascending=False)

# Mostrar gràfic de barres de la importància de les variables
plt.figure(figsize=(10, 15))
sns.barplot(x='importancia', y='feature', data=df_importancia, palette='viridis')
plt.title('Global Feature Importance (sobre tot el dataset)')
plt.xlabel('Importancia')
plt.ylabel('Features')
plt.show()
```

9.7.4 XGBoost - Entrenament

```
from xgboost import XGBClassifier
from sklearn.model_selection import RandomizedSearchCV, StratifiedKFold,
↪  train_test_split
from scipy.stats import randint, uniform
from sklearn.metrics import (
    classification_report, confusion_matrix, roc_curve, auc,
    precision_score, recall_score, f1_score, accuracy_score
)
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np

# -------------------
# Dades d'entrada
# -------------------
path_excel = 'RandomForestTrain.xlsx'
df = pd.read_excel(path_excel)

no_features = ['ensayo_id', 'class', 'Unnamed: 0']
feature_cols = df.columns.difference(no_features)
df[feature_cols] = df[feature_cols].apply(pd.to_numeric, errors='coerce')

X = df[feature_cols]
y = df['class']

# -------------------
# Parametrització
# -------------------
xgb = XGBClassifier(
    use_label_encoder=False,
    eval_metric='logloss',
    random_state=42
)

param_dist = {
    'n_estimators': randint(100, 300),
    'max_depth': randint(2, 30),
    'learning_rate': uniform(0.01, 0.3),
    'subsample': uniform(0.05, 1),
    'colsample_bytree': uniform(0.05, 1),
    'min_child_weight': randint(1, 20),
    'gamma': uniform(0, 0.7),
    'scale_pos_weight': [1.0, (y == 0).sum() / (y == 1).sum()]  # per balancejar
    ↪  si cal
}
```

```
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

random_search = RandomizedSearchCV(
    xgb,
    param_distributions=param_dist,
    n_iter=30,
    scoring='f1',
    cv=cv,
    verbose=2,
    n_jobs=-1,
    random_state=42
)

random_search.fit(X, y)

print("\nMillors hiperparàmetres trobats:")
print(random_search.best_params_)

best_model = random_search.best_estimator_

# --------------------
# Avaluació final
# --------------------
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
↪  stratify=y, random_state=42)

best_model.fit(X_train, y_train)

y_pred = best_model.predict(X_test)
y_pred_proba = best_model.predict_proba(X_test)[:, 1]
```

9.7.5 LightGBM - Entrenament

```
from lightgbm import LGBMClassifier
from sklearn.model_selection import StratifiedKFold, RandomizedSearchCV
from sklearn.utils.class_weight import compute_class_weight
import numpy as np

# Carregar fitxer
path_excel = 'RandomForestTrain.xlsx'
df = pd.read_excel(path_excel)

# Preparar dades
no_features = ['ensayo_id', 'class', 'Unnamed: 0']
feature_cols = df.columns.difference(no_features)
df[feature_cols] = df[feature_cols].apply(pd.to_numeric, errors='coerce')

X = df[feature_cols]
y = df['class']

# Calcular class_weights manualment
classes = np.unique(y)
weights = compute_class_weight(class_weight='balanced', classes=classes, y=y)
class_weights = dict(zip(classes, weights))

# Definir el model
lgbm = LGBMClassifier(random_state=42, class_weight=class_weights, n_jobs=-1)

# Espai d'hiperparàmetres
param_dist_lgbm = {
    'n_estimators': [100, 200, 300],
    'max_depth': [-1, 5, 10, 15, 20],
    'learning_rate': [0.01, 0.05, 0.1, 0.2],
    'num_leaves': [15, 31, 63, 127],
    'min_child_samples': [10, 20, 30, 40],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0]
}

# Configurar cross-validation
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Randomized Search
random_search_lgbm = RandomizedSearchCV(
    lgbm,
    param_distributions=param_dist_lgbm,
    n_iter=30,
    scoring='f1',
    cv=cv,
```

```
    random_state=42,
    verbose=2,
    n_jobs=-1
)

# Ajustar
random_search_lgbm.fit(X, y)

# Millors hiperparàmetres
print("Millors hiperparàmetres LightGBM:")
print(random_search_lgbm.best_params_)
```

9.7.6 Logistic Regression - Entrenament

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import uniform, randint
from sklearn.linear_model import LogisticRegression


# Carregar fitxer
path_excel = 'RandomForestTrain.xlsx'
df = pd.read_excel(path_excel)

# Filtrar el DataFrame
df = df[~df['ensayo_id'].isin(assaigs_exclosos)]


# Preparar dades
no_features = ['ensayo_id', 'class', 'Unnamed: 0']
feature_cols = df.columns.difference(no_features)

df[feature_cols] = df[feature_cols].apply(pd.to_numeric, errors='coerce')

df=df.dropna()

X = df[feature_cols]
y = df['class']

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

param_dist_lr = {
    'C': uniform(0.01, 300),
    'penalty': ['l2'],
    'solver': ['lbfgs']
}

lr = LogisticRegression(max_iter=1000)
random_search_lr = RandomizedSearchCV(lr, param_distributions=param_dist_lr,
↪  cv=5, scoring='f1', n_iter=30, random_state=42)
random_search_lr.fit(X_train_scaled, y_train)
```

9.7.7 Support Vector Machines - Entrenament

```
from sklearn.svm import SVC

# Definim el model i la cerca d'hiperparàmetres
svm = SVC(probability=True, random_state=42)

param_dist = {
    'C': uniform(0.1, 10),
    'gamma': ['scale', 'auto'],
    'kernel': ['rbf', 'poly', 'sigmoid']
}

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

random_search_svm = RandomizedSearchCV(
    svm,
    param_distributions=param_dist,
    n_iter=20,
    scoring='f1',
    cv=cv,
    random_state=42,
    verbose=1,
    n_jobs=-1
)

random_search_svm.fit(X_scaled, y)

best_svm = random_search_svm.best_estimator_
best_svm.fit(X_train, y_train)
```

9.7.8 K-Nearest Neighbours - Entrenament

```
from sklearn.neighbors import KNeighborsClassifier

param_grid_knn = {
    'n_neighbors': randint(1, 10),
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan']
}

knn = KNeighborsClassifier()
random_search_knn = RandomizedSearchCV(knn, param_distributions=param_grid_knn,
↪  cv=5, scoring='f1', n_iter=30, random_state=42)
random_search_knn.fit(X_train_scaled, y_train)
```

9.7.9 Multilayer Perceptron - Entrenament

```
from sklearn.neural_network import MLPClassifier
param_grid_mlp = {
    'hidden_layer_sizes': [
        (50,), (100,),
        (50, 50), (100, 50),
        (100, 100), (50, 100),
        (10, 100), (50, 100), (100, 100), (150, 100),  # fixem 100 a la 2a capa
        (100, 10), (100, 50), (100, 100), (100, 150)   # fixem 100 a la 1a capa
    ],
    'activation': ['relu', 'tanh'],
    'solver': ['adam'],
    'alpha': uniform(0.0001, 0.001),
    'learning_rate': ['constant', 'adaptive']
}

mlp = MLPClassifier(max_iter=1000, random_state=42)
random_search_mlp = RandomizedSearchCV(mlp, param_distributions=param_grid_mlp,
↪  cv=5, scoring='f1', n_iter=30, random_state=42)
random_search_mlp.fit(X_train_scaled, y_train)
```

9.8 Èina complementària: *Error Log Tool*

```
Aquest contingut ha estat retirat per confidencialitat
```

Referències

[1] Applus+ idiada official website. https://www.applusidiada.com/. Accessed: 2025-05-28.

[2] Matplotlib: Visualization with python. https://matplotlib.org/. Accessed: 2025-05-28.

[3] Numpy: Fundamental package for scientific computing with python. https://numpy.org/. Accessed: 2025-05-28.

[4] Pandas: Python data analysis library. https://pandas.pydata.org/. Accessed: 2025-05-28.

[5] Python official website. https://www.python.org/. Accessed: 2025-05-28.

[6] Scikit-learn: Machine learning in python. https://scikit-learn.org/. Accessed: 2025-05-28.

[7] Seaborn: Statistical data visualization. https://seaborn.pydata.org/. Accessed: 2025-05-28.

[8] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. Accessed: 2025-04-22.

[9] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. Accessed: 2025-05-12.

[10] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006. Accessed: 2025-05-05.

[11] Evelyn Fix and Joseph L Hodges Jr. Discriminatory analysis. nonparametric discrimination: Consistency properties. 1951. Accessed: 2025-05-13.

[12] John E Grayson. *Python and Tkinter programming*. Manning Publications, 2000. Accessed: 2025-05-22.

[13] David W Hosmer, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*. John Wiley & Sons, 2013. Accessed: 2025-05-12.

[14] Ben Hoyt and contributors. watchdog: Filesystem events monitoring. https://github.com/gorakhargosh/watchdog, 2024. Accessed: 2025-05-24.

[15] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008. Accessed: 2025-04-21.

[16] Microsoft. Visual studio code. https://code.visualstudio.com/, 2024. Accessed: 2025-05-24.

[17] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. Accessed: 2025-05-13.

[18] Zitao Shen. Machine learning 101: Cross validation. https://zitaoshen.rbind.io/project/machine_learning/machine-learning-101-cross-vaildation/, 2020. Accessed: 2025-05-10.