Versió v1.0

Generalitat de Catalunya

Guia per al desenvolupament de casos d'ús de Machine Learning (ML)

Última actualització gener 2026

Finançat per
Unió Europea
Fons Europeu
Next Generation

GOBIERNO DE ESPAÑA

Plan de Recuperación, Transformación y Resiliencia

Next Generation Catalunya

C. del Foc, 57, edifici B
08038 Barcelona
Tel. 93 857 40 00

Índex

Finançat per
Unió Europea
Fons Europeu
Next Generation
GOBIERNO DE ESPAÑA
Plan de Recuperación, Transformación y Resiliencia
Next Generation Catalunya

C. del Foc, 57, edifici B
08038 Barcelona
Tel. 93 857 40 00

Llista d'abreviatures

Abreviatura	Significat
MLOps	Operacions de Machine Learning
CI	Integració Contínua
CD	Desplegament Continu
DES	Desenvolupament
PRE	Preproducció
PRO	Producció
DAB	Databricks Asset Bundles
ML	Machine Learning
WSL	Subsistema de Windows per a Linux
Workflow	Flux de treball
Workspace	Espai de treball
Features	Característiques
Schedule	Programació

1. Introducció

La següent documentació proporciona una guia per a la implementació de casos d'ús de ML de classificació, regressió i predicció dins la plataforma MLOps desenvolupada, allotjada a Azure Databricks. S'expliquen les millors pràctiques de Databricks per a científics de dades, enginyers de ML i equips de DevOps que busquen optimitzar tot el cicle de vida dels models de ML. A més, defineix un procés estructurat de desplegament, des del desenvolupament fins als entorns de preproducció i producció.

1.1 Visió general de l'arquitectura aplicada a MLOps

La gestió del cicle de vida d'un **cas d'ús de Machine Learning (ML)** requereix organitzar i mantnir diferents entorns. Aquests entorns ajuden a estructurar el desenvolupament, les proves i la implementació de models d'aprenentatge automàtic alhora que garanteixen l'escalabilitat, la governança i el control de costos. L'arquitectura definida a Azure Databricks permet els punts següents.

- **Segmentació d'entorns**
    - El **desenvolupament**, les **proves** i la posada en **producció** se separen en entorns diferents: **desenvolupament (DES)**, **preproducció (PRE)** i **producció (PRO)**.
    - Això ajuda a **aïllar els problemes** abans de la implementació a producció.
- **Gestió del cicle de vida del model**
    - **Repositori de codi i pipelines de CI/CD:** El proveïdor utilitzarà un repositori per al projecte o cas d'ús, que tindrà configurades les *pipeline*s de CI/CD que permetran fer el pas entre entorns.
    - **Entorn de desenvolupament (DES):** Dins de l'entorn de DES, el proveïdor desenvoluparà el model de ML a l'espai de treball corresponent al domini d'informació del projecte o cas d'ús.
    - **Entorn de preproducció (PRE):** Una vegada el model està desenvolupat, s'utilitzen les *pipeline*s de CI/CD per fer les proves sobre l'entorn de PRE. Les proves unitàries es realitzaran sobre les *pipeline*s de CI/CD i les proves d'integració s'executaran sobre l'espai de treball (worksapce) de l'entorn de PRE corresponent al domini d'informació del projecte o cas d'ús. En aquest entorn es realitzaran totes les validacions necessàries abans de passar a producció.
    - **Entorn de producció (PRO):** Un cop passades les proves i les validacions, s'utilitzen les *pipeline*s de CI/CD per passar el codi a PRO, dins l'espai de treball corresponent al domini d'informació del projecte o cas d'ús.
- **Gestió de Costos i Recursos**

- Cada **proveïdor** tindrà un espai de treball separat a l'entorn de **Desenvolupament** per garantir un seguiment clar dels costos.
- Els espais de treball estan alineats amb els dominis d'informació de la Generalitat de Catalunya, garantint una governança eficient i una òptima utilització dels recursos.

Aquest enfocament estructurat garanteix que els projectes ML segueixin les millors pràctiques d'**escalabilitat, seguretat i governança,** alhora que dona suport a una transició suau del **desenvolupament al desplegament**.

1.2 Visió general de MLOps

Per tal d'executar i estandarditzar tots els processos de MLOps, s'ha desenvolupat un codi font que servirà com a plantilla i que estarà accessible en un repositori.

Aquest document se centra en l'ús integral d'aquest codi font, des del moment en què es clona des del repositori fins al desenvolupament a l'entorn DES i el posterior desplegament en PRE i PRO. Es proporcionen instruccions pas a pas per a:

- Configurar l'entorn de DES, cobrint totes les dependències, configuracions i permisos necessaris.
- Adaptar la plantilla de codi per satisfer les necessitats específiques dels casos d'ús mitjançant els Databricks Asset Bundles (DABs).
- Desplegar els models en entorns de preproducció i producció, automatitzant les *pipeline*s per a la integració contínua (CI) i el desplegament continu (CD).
- Establir els processos de monitoratge i avaluació per garantir que els models es mantinguin robustos i fiables al llarg del temps.

Aquesta guia té com a objectiu explicar el procés d'integració dins de la plataforma MLOps, permetent als usuaris construir, desplegar i mantenir solucions ML escalables i d'alt rendiment amb facilitat, a partir d'un codi de plantilla. El proveïdor només gestiona l'espai de treball de

DES. Els altres entorns i espais de treball (PRE i PRO) són gestionats pel proveïdor de la PTD, i les *pipelines* de CI/CD pel SIC+.

A continuació es mostra el diagrama amb els passos a seguir per a desenvolupar, testejar i desplegar un projecte de ML. Aquest procés de MLOps pot estar subjecte a canvis, per exemple, es preveu l'ús del SIC+ o la lectura de dades de PRO.

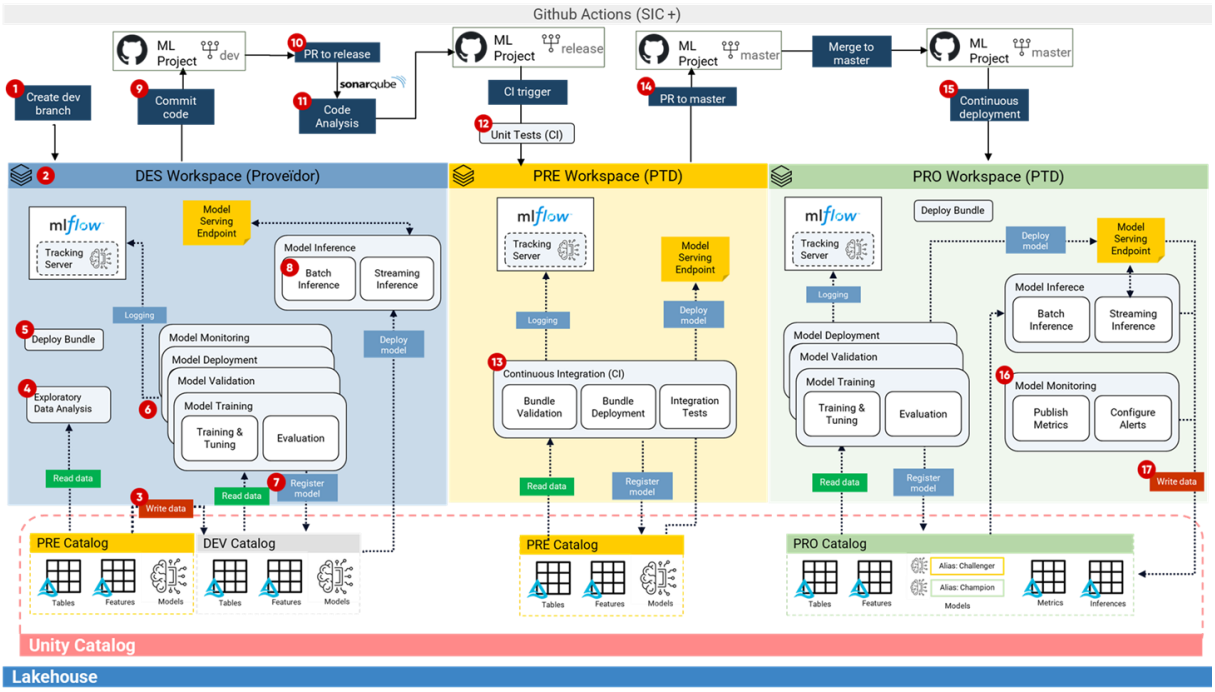

Figura 1 – Procés de MLOps entre entorns DES, PRE i PRO juntament amb el repostori git i els processos CI/CD

Pas	Descripció
1	Cloneu la plantilla de ML des del repositori a la vostra branca de desenvolupament. Modifiqueu el codi per alinear-lo amb el cas d'ús específic. Més detalls a l'apartat: ¡Error! No se encuentra el origen de la referencia.
2	Assegureu-vos que el vostre entorn estigui configurat segons les directrius establertes. Autentiqueu el CLI de Databricks per habilitar l'accés a l'espai de treball. Seguint els passos de les **seccions** ¡Error! No se encuentra el origen de la referencia. i ¡Error! No se encuentra el origen de la referencia.
3	Si cal, llegiu les dades de l'entorn PRE per realitzar EDA i entrenar els vostres models en entorn DES. Per obtenir-lo, envieu una sol·licitud al proveïdor de la PTD seguint els passos de l'apartat ¡Error! No se encuentra el origen de la referencia..

4	Realitzeu l'anàlisi exploratòria de dades (EDA), la selecció del model i els processos d'enginyeria de característiques. Abans del pas 4, hauríeu d'haver definit el model que voleu implementar, juntament amb els seus hiperparàmetres, mètriques d'avaluació i features d'entrada..
5	En aquest pas, configureu el codi de la plantilla DAB en funció del cicle de vida del model i de les definicions de feature engineering del pas 4. Implementeu-lo a l'àrea de treball del DES juntament amb els fluxos de treball i els recursos configurats. Per a més detalls, consulteu Recursos del Databricks. **Nota**: A l'entorn DES, és probable que aquest pas es repeteixi diverses vegades a mesura que itereu en els fluxos de treball. Els passos següents descriuen les tasques necessàries per a cada iteració de l'espai de treball.
6	En primer lloc, creeu els recursos de dades (esquemes, taules i volums) que necessiteu executant l'espai de treball detallat a la secció Crea un flux de treball d'objectes. Apliqueu els vostres processos de feature engineering mitjançant l'espai de treball detallat a la secció **Flux de treball de Feature Engineering.** Entrena i avalua el model amb els features. Tot i que és opcional, l'aprofitament del feature store de Databricks pot simplificar el seguiment del llinatge entre dades i models. Valideu el model i continueu amb la implementació. Configureu les mètriques d'avaluació per a la supervisió contínua. Consulteu els detalls de l'espai de treball a la secció **Flux de treball de monitoratge.**
7	Quan el model funciona bé i ha superat el pas d'avaluació. Podeu registrar la nova versió del model (si el model ja existia) al **Desenvolupament (DES).**
8	Modificar el codi del flux de treball (workflow) d'inferència per lots a la secció per alinear-se amb les dades que voleu inferir. Seleccioneu les dades adequades de les taules i apliqueu els processos de feature engineering necessaris per generar l'entrada del model.
9	Després de completar l'entrenament, la validació, la implementació i la supervisió del model i la inferència, assegureu-vos que les configuracions del flux de treball estiguin finalitzades al codi del bundle. A continuació, envieu el codi a la branca de desenvolupament del vostre repositori.
10	Quan estigueu preparats per a les proves, pugeu el codi a la branca principal mitjançant una pull request (PR) a **Repositori Git & Pipelines CI/CD.**
11	Utilitzeu el pipeline del flux de treball detallat a la secció **Repositori Git & Pipelines CI/CD** per dur a terme l'anàlisi de codi amb **SonarQube** i les proves corresponents.
12	En aquest pas, s'executen les proves unitàries configurades prèviament. **Nota**: Si us plau, consulteu més detalls sobre les proves unitàries a la secció: Proves unitàries.
13	També s'executen les proves d'integració que has configurat prèviament a DES. Aquestes proves s'executen abans de desplegar el bundle a l'espai de treball (worksapce) PRE des de PTD.

	Consulta amb el proveïdor de la PTD els requisits per migrar a PRE i adapta el codi en conseqüència (especialment les variables del fitxer databricks.yml). **Nota**: Si us plau, consulteu més detalls sobre les **proves** d'integració a la secció: Proves d'integració
14	Després de l'execució exitosa de totes les proves i la validació a la canalització CI/CD de PRE, la Pull Request (PR) a la branca principal (master) s'hauria d'aprovar i el codi del bundle es combina a la branca principal.
15	Consulteu amb el proveïdor de la PTD els requisits per migrar a PRO i adapteu el codi en conseqüència (especialment les variables del fitxer Databricks.yml). El proveïdor de la PTD publica el codi validat i el desplega a l'espai de treball PRO des de PTD. Un cop la migració del codi és exitosa, el bundle es desplega a **Producció (PRO).**
16	Quan es treballa en producció, és important monitorar el model utilitzant els fluxos de treball definits al Pas 6. Algunes mètriques s'emmagatzemen per defecte, però també pots definir altres mètriques que consideris necessàries.
17	Analitzeu les mètriques del pas anterior per garantir el rendiment i l'estabilitat del model a l'entorn de producció (per exemple, totes les inferències del model, mètriques d'entrenament i estadístiques de dades d'entrada).

2. Primers passos

2.1 Prerequisits

Aquest projecte té diversos requisits previs essencials per a un funcionament adequat. A més, per implementar recursos al Databricks, necessitareu una instal·lació de la CLI del Databricks, així com els permisos adequats de l'Azure per interactuar amb les àrees de treball del Databricks necessàries.

2.1.1 Databricks CLI

Per a aquest projecte, instal·leu la versió 0.205.0 o posterior de la CLI del Databricks. Suggerim dos enfocaments, però altres es poden trobar a Install or update the Databricks CLI - Azure Databricks | Microsoft Learn.

Windows:

Des del vostre Command Prompt, executeu les dues ordres winget següents per instal·lar el CLI i, a continuació, reinicieu el Command Prompt:

winget search databricks

winget install Databricks.DatabricksCLI

Per finalitzar correctament la instal·lació, reinicieu el terminal on estàveu treballant. Confirmeu si la CLI del Databricks està instal·lada correctament. Per fer-ho, visualitzeu la versió de l'executable de la CLI de Databricks mitjançant l'opció -v o executeu el comendament version:

databricks -v

Linux:

Comenceu instal·lant el subsistema de Windows per a Linux (WSL) des de Microsoft Store, concretament la versió Ubuntu 22.04.3 LTS. Un cop configurat WSL, utilitzeu les ordres següents per instal·lar la CLI del Databricks. En aquest document, la instal·lació es fa mitjançant Homebrew:

brew tap databricks/tap

brew install databricks

Si Homebrew encara no està instal·lat al vostre entorn WSL, podeu instal·lar-lo mitjançant l'ordre següent:

/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

Després de la instal·lació, assegureu-vos que la CLI del Databricks estigui actualitzada:

brew upgrade databricks

2.1.2 Autenticació Databricks

A continuació, autentiqueu la CLI del Databricks per habilitar la implementació de recursos a l'àrea de treball del Databricks de desenvolupament designada.

Comenceu generant un token d'accés personal. Aneu al vostre espai de treball (worksapce)del Databricks i feu clic al cercle situat a la cantonada superior dreta (Figura 3 configuració de l'usuari de la interfície d'usuari de l'àrea de treball del Databricks).

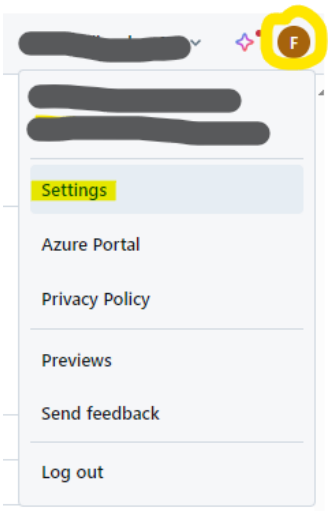

Figura 2 configuració de l'usuari de la interfície d'usuari de l'espai de treball del Databricks

A continuació, feu clic a **Configuració**, seleccioneu **Desenvolupador** i després **Tokens** d'accés, i feu clic a **Gestionar**. Creeu un nou **token** i **emmagatzemeu**-lo de manera segura. (Figure 3)

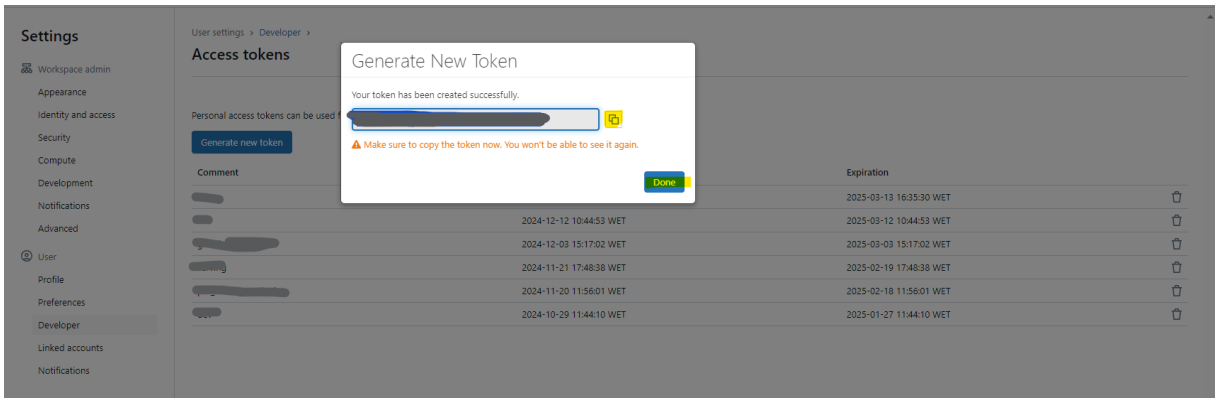

Figure 3 – Generació d'un nou token a la interfície de Databricks

Al vostre terminal, executeu l'ordre següent per configurar la CLI del Databricks:

databricks configure

Quan se us sol·liciti, haureu de disposar de la informació següent:

- **Databricks Host** - correspon a l'adreça URL de l'àrea de treball de Databricks.
- **Token d'accés personal** - proporcioneu el token d'accés personal que acabeu de generar.

Per confirmar que el vostre perfil s'ha creat correctament, executeu el codi següent:

databricks auth profiles

I assegureu-vos de validar el vostre perfil amb SI:

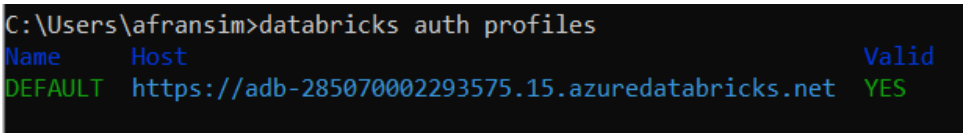

Figura 4 - Perfil de Databricks per defecte validat a l'espai de treball

Això completa el procés de configuració, permetent un desplegament i gestió fluid dels recursos de Databricks. Un cop finalitzat, el perfil de configuració corresponent es desarà al vostre fitxer .databrickscfg, i podreu desplegar els recursos de Databricks a l'espai de treball de destinació.

Per obtenir informació més detallada, consulteu https://learn.microsoft.com/en-us/azure/databricks/dev-tools/auth/azure-cli.

2.2 Instruccions de configuració

Després de configurar els requisits previs del mòdul anterior, ara es realitza la configuració del projecte de plantilla ML per crear el vostre cas d'ús personalitzat. El primer pas és **clonar** el repositori de plantilles.

Per accedir al repositori de plantilles:

- Els usuaris han de tenir el correu electrònic de GICAR.
- Aneu a https://github.com/ctti-dev/3670.00-sta-ml_template-databricks/tree/feature/template i sol·liciteu accés als usuaris.

Per sol·licitar accés a aquest repositori, contacteu amb:

- S'ha de demanar a través de **JIRA** amb el servei de **ACOCLDSIC**

Després d'obtenir els permisos d'accés necessaris, podeu clonar el repositori de plantilles a la vostra màquina local.

1. Copieu l'URL del repositori proporcionat per a la plantilla.
2. Aneu al directori desitjat a la vostra màquina local on voleu clonar el repositori.
3. Executa la següent ordre al terminal:

   git clone https://github.com/ctti-dev/3670.00-sta-ml_template-databricks.git

En cas que trobeu un error de certificat SSL, podeu solucionar-lo temporalment amb l'ordre:

git -c http.sslVerify=false clone https://github.com/ctti-dev/3670.00-sta-ml_template-databricks.git

Nota:

- *L'URL del repositori proporcionat està destinat únicament a accedir a la plantilla.*
    - *Durant el desenvolupament del vostre cas d'ús específic, haureu de crear i utilitzar el vostre propi repositori al SIC+ (s'ha de demanar a través de **JIRA** amb el servei de **ACOCLDSIC**).*

2.3 Estructura del codi

L'estructura del codi segueix l'enfocament de MLOps Stacks del Databricks, aneu a MLOps Stacks: model development process as code - Azure Databricks | Microsoft Learn com a referència. Després de clonar el repositori de plantilles MLOps, el directori hauria de tenir la carpeta arrel ntt_sta_ml_template i s'hauria d'organitzar de la següent manera:

- **.gitignore:** Aquest fitxer especifica quins fitxers i directoris Git hauria d'ignorar quan faci el seguiment dels canvis. Normalment, inclou fitxers temporals, dades sensibles, registres i configuracions d'entorn que no s'han de versionar.

- **README.md**: El fitxer de documentació principal del projecte. Proporciona una visió general del projecte, instruccions de configuració, directrius d'ús i qualsevol altra informació rellevant per als usuaris i col·laboradors

- **ml_template:** Un directori de la plantilla que serà reemplaçat pel nom real del projecte. Conté tots els fitxers i subdirectoris específics del codi Databricks Asset Bundles.
    - **databricks.yml**: Fitxer arrel del *bundle* per al projecte ML que defineix la configuració del *bundle*, nom, recursos, variables, espais de treball de destinació, etc. Aquest fitxer s'ha d'editar per assignar valors a cada variable dinàmica. Cada variable es pot configurar per tenir diferents valors segons l'entorn (DES, PRE o PRO).
    - **test-requirements.txt**: Una llista de dependències requerides específicament per a executar les proves d'integració i unitat. Assegura que s'instal·len les biblioteques i eines correctes per validar la funcionalitat del projecte.
    - **create_objects/:** Aquest directori conté els *script*s per crear els objectes de dades necessaris a les ubicacions d'emmagatzematge adequades.

- **notebooks/CreateObjects.py**: Llibreta responsable de la creació de tots els objectes de dades.

- **deployment/:** Aquest directori conté *script*s relacionats amb el desplegament de models i l'execució d'inferències.
  - **batch_inference/:** Conté recursos i *script*s per realitzar inferències per lots, on el model processa diversos punts de dades alhora.
    1. **predict.py:** Manté la funció responsable de processar les dades d'entrada per lots i generar prediccions basades en el model i les dades d'entrada i desar les prediccions a Unity Catalog. Aquest fitxer importa funcions d'enginyeria de característiques (*feature*s) del fitxer d'utilitat d'enginyeria de característiques, que s'ha d'adaptar d'acord amb la necessitat.
    2. **notebooks:**
       i. **.codecarbon.config:** Fitxer de configuració per al rastrejador d'emissions CodeCarbon.
       ii. **BatchInference.py:** Aquest *notebook* està dissenyat per realitzar inferències per lots mitjançant un model registrat. Valida la convenció de nomenclatura de variables, recupera el model campió del registre de models MLflow i executa l'*script* de predicció per lots.
  - **model_deployment/:** Inclou els *script*s per desplegar models en entorns de producció.
    1. **deploy.py:** Aquest *script* automatitza el desplegament d'un model registrat per MLflow assignant-li l'àlies apropiat ("Champion") al Catàleg d'unitats. Pren un model d'URI i entorn (DES, PRE o PRO) com a entrades, assegurant que la versió del model especificada estigui marcada com a model preparat per a la producció.
    2. **notebooks/ModelDeployment.py:** Aquest *notebook* automatitza el desplegament d'un exemple de regressió ML o model de classificació.

- **feature_engineering/:** Conté *script*s per a l'enginyeria de característiques.
  - **utils.py:** Manté totes les funcions de processament i transformació de dades, per preparar les dades en brut en característiques (*feature*s) per al consum de models. Aquest fitxer s'ha d'editar, ajustar i afegir funcions segons les necessitats i especificitats de cada cas d'ús.
  - **notebooks/FeatureEngineering.py:** Aquest *notebook* automatitza la preparació de dades, ingerint dades en brut, transformant i generant característiques (*feature*s) a guardar en taules per a l'entrenament i validació de models.

- **monitoring/**: Aquest directori conté eines per supervisar els models desplegats, assegurant-se que funcionen com s'esperava en producció.

- **utils.py:** Conté funcions d'utilitat per donar suport a tasques de supervisió.
- **notebooks:**

    1. **CreateMonitoringTable.py:** Aquest *notebook* de Databricks està enfocat a instanciar una taula que conté informació de model i dades d'inferència amb finalitats de monitorització.
    2. **CreateQualityMonitor.py:** Aquest *notebook* Databricks està dissenyat per instanciar un Monitor de Qualitat per avaluar el rendiment del model.
    3. **CreateMonitoringAlerts.py:** Aquest *notebook* Databricks està dissenyat per crear consultes i alertes per avaluar el rendiment del model.

4. **resources/:** Emmagatzema els fitxers de configuració YAML necessaris per configurar els fluxos de treball del Databricks

    - **batch-inference-workflow-resource.yml**: Consulta els detalls a l'apartat Flux de treball Batch Inference.
    - **create-objects-workflow-resource.yml**: Consulta els detalls a l'apartat Crea un Flux de treball d'objectes.
    - **feature-engineering-workflow-resource.yml**: Consulta els detalls a l'apartat Feature Engineering workflow.
    - **model-workflow-resource.yml:** Consulta els detalls a l'apartat Flux de treball del model.
    - **monitoring-resource.yml:** Consulta els detalls a l'apartat Flux de treball de monitoratge.

- **serving/:** Conté fitxers relacionats amb el *Model Serving* per a realitzar la inferència en temps real o en línia.

    - **utils.py:** Aquest *script* d'utilitat inclou mètodes per comprovar si existeix un *serving endpoint*, crear o actualitzar els punts finals i esperar que estiguin llestos. Les funcions recuperen i comparen les configuracions dels punts finals, incloses les entitats servides, les configuracions del trànsit i les opcions de captura automàtica. L'*script* comprova si un punt final requereix actualitzar comparant les configuracions actual i nova. Gestiona les sol·licituds HTTP a l'API de Databricks per a aquestes operacions i inclou la gestió d'errors i el registre per garantir una execució suau dels punts finals de servei per als models ML. Aquest *script* conté les següents funcions/mètodes d'ajuda.

    - **notebooks/ModelServing.py:** Aquesta llibreta configura o actualitza un *serving endpoint* del model Databricks, mitjançant les funcions auxiliars definides a l'*script* anterior.

- **tests/:** Aquest directori conté proves unitàries, proves d'integració i registres per garantir la funcionalitat del projecte tal com es pretén.

- **__init__.py:** (Buit) fitxer d'inicialització, necessari perquè les proves s'executin com s'esperava.
- **integration_tests/:** Proves que validen la integració de múltiples components, assegurant que funcionen junts com s'esperava.
- **unit_tests/:** Prova les funcions o components individuals, assegurant que cada part del sistema funciona de forma aïllada.

○ **training/:** Conté *script*s per entrenar models d'aprenentatge automàtic.

- **utils.py:** Un *script* d'utilitat que conté funcions d'ajuda per a la preparació de dades i models d'entrenament.
- **notebooks:**
  1. **.codecarbon.config:** Fitxer de configuració per al rastrejador d'emissions CodeCarbon.
  2. **ModelTraining.py:** Aquest *notebook* d'exemple automatitza l'entrenament d'un model XGBoost per a tasques de regressió. Valida la convenció de noms de variables, ingereix característiques, divideix dades i entrena el model, registrant-lo al registre MLflow.

○ **validation/:** Aquest directori conté *script*s i eines per validar els models abans que es despleguin.

- **utils.py:** Aquest fitxer defineix diverses funcions d'ajuda per gestionar i registrar detalls relacionats amb les carreres d'entrenament de models en MLflow.

- **validation.py:** Aquest fitxer defineix funcions per a mètriques personalitzades, llindars de validació i configuració de l'avaluador en el context de l'avaluació del model MLflow. És important tenir en compte que les funcions descrites a continuació, incloses dins del fitxer validation.py, han de ser retocades per l'equip de Data Science d'acord amb el model de ML en qüestió. Les mètriques personalitzades, el llindar de validació i *evaluator_config* són funcions utilitzades com a paràmetre en l'avaluació Mlflow, encara per configurar, referides a mlflow documentation.

  1. **custom_metrics:** Defineix i retorna una llista de mètriques personalitzades per a l'avaluació de models. Actualment inclou una funció mètrica personalitzada, squared.diff.plus.one, que calcula la diferència al quadrat entre les columnes "predicció" i "objectiu", ajustada afegint-ne una. Aquesta mètrica està configurada per a afavorir valors més baixos.
  2. **validation_thresholds**: Estableix un diccionari de regles de validació utilitzant MetricThreshold. Actualment inclou condicions per a maxerrorerror, que ha de ser menor o igual a 500, i mean,squared,error, que també ha de ser menor o igual a 500,

assegurant que ambdues mètriques indiquen un rendiment de model acceptable.

3. **evaluator_config**: Proporciona un diccionari de configuració per a l'avaluador. En la implementació actual, retorna un diccionari buit, indicant que no s'han establert configuracions d'avaluador específiques. Això es pot modificar segons sigui necessari per a una personalització posterior de l'avaluació.

- **notebooks:**
- **.codecarbon.config:** Fitxer de configuració per al rastrejador d'emissions CodeCarbon.
    1. **ModelValidation.py:** Aquest *notebook* utilitza les funcions prèviament definides per a la validació de models en un entorn Databricks, aprofitant el flux ML per al seguiment i la gestió de models.

2.4 Configuració

El primer pas per configurar el projecte segons el vostre cas d'ús és configurar el *fitxer* [1]databricks.yml.

Aquest fitxer serveix com un fitxer de configuració que configura fluxos de treball, clústers, permisos i espais de treball. Cada variable s'ha de substituir pels espais de treball PRE i PRO segons les seves especificitats. Cal garantir que tots els valors estiguin correctament poblats per poder realitzar una generació del projecte adequada.

A continuació es mostra la llista de variables incloses en *databricks.yml* fitxer:

- **catalog:** catàleg on es guarden les dades d'entrenament al Catàleg Unity.
- **managed_location:** ubicació a l'emmagatzematge al núvol que s'utilitza per emmagatzemar catàlegs, esquemes i taules.
- **external_location:** ubicació a l'emmagatzematge al núvol que s'utilitza per emmagatzemar els volums.
- **finops_projecte**: s'utilitza per assignar clústers i altres recursos per a la segregació de costos. Consultar amb proveïdor de la PTD com han de taggejar el FinOps i indicar-ho.
- **model_name:** nom que s'utilitzarà per al model ML.
- **use_case:** nom del cas d'ús, utilitzat per anomenar tots els recursos relacionats amb el cas d'ús. Consultar amb PTD com han anomenar.
- **notification_email:** correu electrònic al qual s'enviaran les alertes quan el flux de treball de Datericks tingui èxit o falli.
- **model_type:** S'utilitza per donar suport a l'avaluació del model mitjançant mètriques personalitzades de MLflow i per donar suport a la supervisió de models

[1] Per obtenir més detalls sobre la configuració del paquet, consulteu [la configuració del paquet d'actius del Databricks - Azure Databricks | Microsoft Learn]

(consulteu els tipus de models de ML disponibles a la documentació oficial https://mlflow.org/docs/latest/python_api/mlflow.html#mlflow.evaluate). Per exemple: 'regressor o 'classificador''

- **run_mode:** Mode d'execució (disabled, dry_run, enabled):
  - 'disabled': No s'ha de fer cap validació abans del desplegament.
  - 'dry_run': Els llindars de validació no s'utilitzaran per fallar la validació general.
  - 'enabled': Els errors de validació del llindar bloquejaran el desplegament del model (*model deployment*).
- **inference_output_table:** nom de la taula on es desarà automàticament la sortida de la inferència.
- **monitoring_workflow_schedule:** "cron expression" que determinarà la periodicitat amb què s'executarà el workflow de monitorització.
- **monitor_baseline:** 'true' si les dades de base/veritat estan disponibles per a la comparació.

2.5 Bones Pràctiques per personalitzar els fluxos de treball de Databricks

La plantilla proporcionada està dissenyada per cobrir els passos més comuns d'un cas d'ús típic de ML, oferint una base sòlida sobre la qual basar-se. No obstant això, cada cas d'ús és únic, i és probable que es requereixi algun nivell de personalització per alinear-se amb els requisits específics o la lògica empresarial. Per assegurar que el vostre flux de treball sigui eficient, mantenible i escalable, us recomanem seguir aquestes bones pràctiques.

2.5.1 Llibreries

Respecte a les llibreries, és recomanable especificar la versió concreta que es fa servir ja que:

- Evita conflictes de dependències entre diferents treballs que s'executen al mateix clúster.
- Redueix el temps d'inici, ja que les biblioteques només s'instal·len quan són necessàries
- Garanteix la reproductibilitat, ja que cada tasca obté les versions exactes de les biblioteques requerides.

Per aconseguir-ho, haureu de definir la versió de les llibreries a instal·lar en la definició del flux de treball utilitzant el fitxer yml específic.

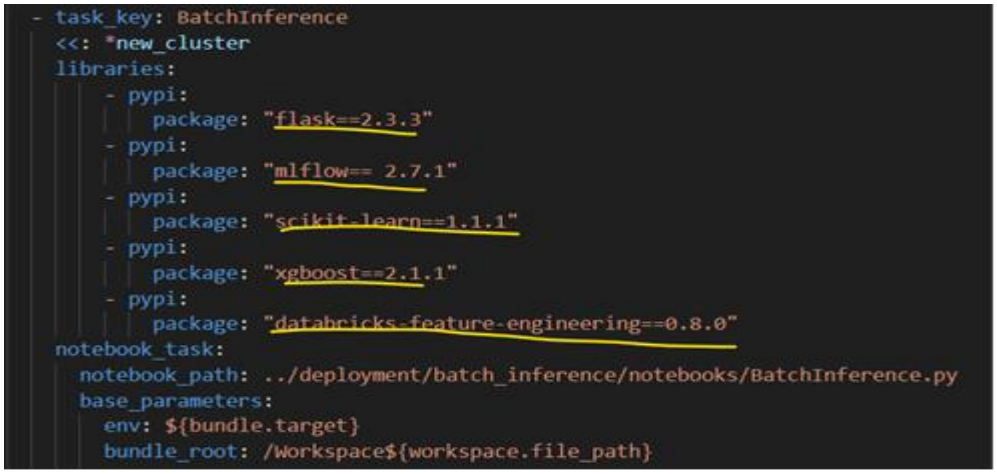

Figura 7 Definició de cinc llibreries a nivell de tasca en el flux de treball de Batch Inference

(batch-inference-workflow-resource.yml)

2.5.2 Centralitzar el codi reutilitzable

Crea arxius `utils.py` per a emmagatzemar totes les funcions auxiliars i reutilitzables. En importar aquest mòdul als vostres *notebook*s o *script*s, elimineu la duplicació de codi i les actualitzacions de la línia de flux de treball.

2.5.3 Parametritza els notebooks

Utilitza `dbutils.widgets` per passar paràmetres dinàmics als *notebook*s. Això fa que els vostres fluxos de treball siguin flexibles i reutilitzables en diferents entorns.

Exemple:

```
dbutils.widgets.text("input_path", "/path/to/data")
input_path = dbutils.widgets.get("input_path")
```

2.5.4 Fluxos de treball modulars

Divideix els teus fluxos de treball en blocs de notes o fitxers Python separats, cadascun responsable d'una funcionalitat específica, com ara la ingesta de dades, la transformació o l'entrenament de models. Això millora la llegibilitat, la depuració i l'escalabilitat.

2.5.5 Control de versions amb Git

Integra el teu projecte Databricks amb un repositori Git per a habilitar el control de versions i la col·laboració. Publica canvis regularment, utilitza branques de funcionalitats per a actualitzacions i assegura que tots els *script*s se sincronitzen amb el repositori.

Per obtenir més detalls sobre com sol·licitar el repositori Git i les *pipeline*s de CI/CD associades, consulteu la secció 5. Repositori Git & Pipelines CI/CD

2.5.6 Recomenacions de desenvolupament

Recomanem utilitzar un Entorn de Desenvolupament Integrat (IDE) com **VSCode** per escriure i gestionar el vostre codi. Aquesta configuració sincronitza el vostre projecte amb el repositori Git, fent que el control de versions i la col·laboració siguin més fluides.

Per a qualsevol fitxer Python vinculat a tasques de treball Databricks, assegureu-vos que la primera línia de cada fitxer inclogui el següent comentari:

```
# Databricks notebook source
```

Això assegura que el fitxer s'interpreta com un *notebook* de Databricks quan s'importa. També fa que la depuració o el treball exploratori siguin més fàcils d'utilitzar dins de la interfície d'usuari de Databricks.

En adherir-se a aquestes pràctiques i aprofitar un IDE, es poden personalitzar els fluxos de treball de manera efectiva mantenint un procés de desenvolupament robust i eficient.

3. Recursos del Databricks

Després de desplegar el codi font de la plantilla, ja es pot consultar tota l'estructura del projecte al fitxer README.md que es troba a la nova carpeta del projecte.

Ara es pot implementar els recursos a l'entorn del Databricks DES. A la línia d'ordres, només cal que executeu l'ordre següent:

```
databricks bundle deploy -p <configuration-profile-name>
```

Això desplegarà els recursos Databricks seguint les instruccions especificades al fitxer databricks.yml. Aquest fitxer especifica els recursos a desplegar a Databricks (dins de la

carpeta de recursos) i els espais de treball de destinació existents (DES, PRE i PRO). En executar aquesta ordre, estarem instanciant els quatre fluxos de treball per a la plantilla.

3.1 Databricks Asset Bundles

Databricks Asset Bundles (DAB)[2] racionalitza la gestió i desplegament de recursos de Databricks, com ara fluxos de treball, clústers, notebooks, etc., empaquetant-les en bundles estructurats. Aquests bundles permeten una integració perfecta amb les pipelines CI/CD.

DAB alinea el desenvolupament d'aplicacions basades en dades amb els principis moderns d'enginyeria de programari, automatitzant els passos clau del cicle de vida que abans requerien un esforç manual. Aquest enfocament estructurat simplifica la governança, millora la fiabilitat i accelera el cicle de vida del desenvolupament, convertint el DAB en una eina essencial per als projectes que utilitzen els Databricks.

Els passos següents descriuen com començar a utilitzar Databricks Asset Bundles a la terminal:

1. **Inicia la gestió de perfil amb Databricks CLI**
   Executeu l'ordre següent per configurar l'autenticació per a cada àrea de treball de destinació:

   databricks auth login –host <workspace-url>

   Seguiu les indicacions per desar la configuració com a perfil de la CLI del Databricks. Premeu Retorn per acceptar el nom de perfil per defecte o especificar un nom personalitzat.

2. **Llista i inspecció de perfils**
   Per visualitzar els perfils existents:

   databricks auth profiles

   Per visualitzar la configuració existent d'un perfil específic, executeu l'ordre:

   databricks auth env –profile <profile-name>

3. **Autenticació completa**
   Al navegador web, seguiu les instruccions en pantalla per iniciar la sessió a l'àrea de treball especificada de l'Azure Databricks.

[2] DAB's Documentació oficial What are Databricks Asset Bundles? - Azure Databricks | Microsoft Learn

4. **Desplegar recursos amb Databricks Asset Bundles**
   Un cop configurada l'autenticació, podeu desplegar recursos a l'espai de treball Databricks DES utilitzant el framework DAB.

   databricks bundle deploy -p <profil·le-name> -t <target>

   Això desplegarà els recursos Databricks seguint les instruccions especificades al fitxer databricks.yml. Aquest fitxer especifica els recursos a desplegar a Databricks (dins de la carpeta de recursos) i els espais de treball de destinació existents (DES, PRE i PRO). En executar aquesta ordre, estarem instanciant els fluxos de treball.

3.2 Fluxos de treball

Per a aquest projecte, hi ha quatre fluxos de treball diferents, configurats mitjançant fitxers YAML, a la carpeta de recursos.

Tot i que la plantilla proporcionada cobreix els passos més habituals en casos d'ús d'aprenentatge automàtic, és possible que el vostre cas d'ús concret requereixi un cert grau de personalització.

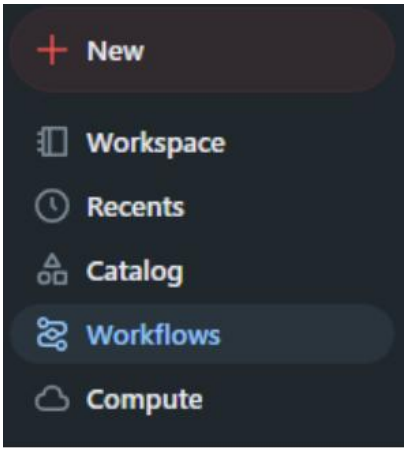

Figura 8: Fluxos de treball d'accés a desplegats al Databricks

Més endavant en aquest capítol documentarem la finalitat i funcionalitats de les tasques, notebooks i proves d'aquest projecte. A la barra lateral esquerra, navegueu a Workflows, després cerqueu els fluxos de treball i executeu-los per esbrinar les configuracions futures necessàries.

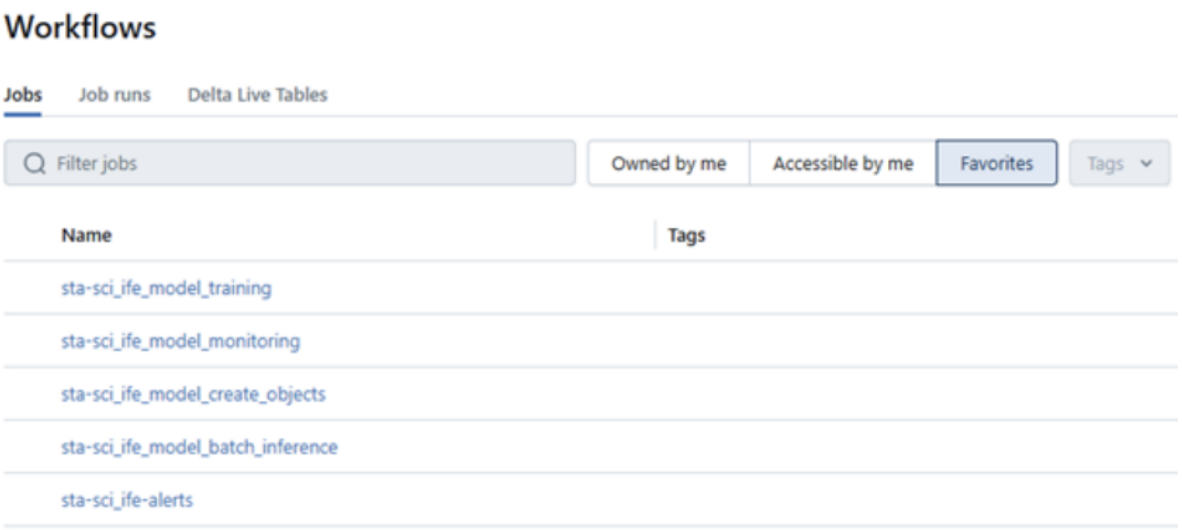

Figura 9 – Flux de treball desplegats pel bundle

3.2.1 Crea un flux de treball d'obejctes

El fitxer create-objects-workflow-resource.yml configura un treball Databricks per crear objectes de dades com esquemes, taules i volums. Defineix un clúster de tasques amb un worker, configuració personalitzada de l'Spark i permisos d'usuari. El treball inclou una única tasca, CreateObjects, amb totes les variables necessàries instanciades dins del fitxer per a l'execució del flux de treball. Primer flux de treball a executar:

Nota: La creació de catàlegs està gestionada pel proveïdor de la PTD.

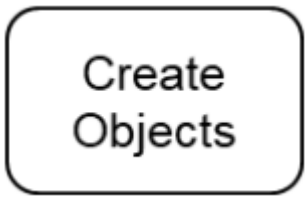

Figura 10 – Crea un diagrama de flux de treball d'objectes

El codi d'aquesta tasca es pot trobar al fitxer:

Create_objects\notebooks\CreateObjects.py

3.2.2 Flux de treball de Feature Engineering

El *feature-engineering-workflow-resource.yml* inclou una tasca associada amb els fitxers de la carpeta **feature_engineering** i implica processar les dades i els features de compilació per servir com a entrades per al model.

Figura 11 – Diagrama del flux de treball de Feature Engineering

El codi d'aquesta tasca es pot trobar al fitxer:

feature_engineering\notebooks\FeatureEngineering.py

Aquest *notebook* està preparat per utilitzar Databricks Feature Store per al seguiment de llinatges i la versió de característiques. Els *features* es guarden en una taula, per al consum en altres tasques.

3.2.3 Flux de treball del model

Aquesta configuració YAML configura un treball de Databricks per a l'entrenament de models, definint un nou clúster amb un worker i configuració específica de Spark, juntament amb permisos per a l'accés de l'usuari. El clúster està etiquetat per al seguiment de FinOps amb un nom de projecte específic.

El treball consisteix en múltiples tasques: entrenament del model (model training), validació del model (model validation), desplegament del model (model deployment) i servei del model (model serving), cadascuna amb dependències i paràmetres definits. Les notificacions per correu electrònic es poden configurar per a l'èxit o fracàs de la feina, assegurant un seguiment adequat del flux de treball.

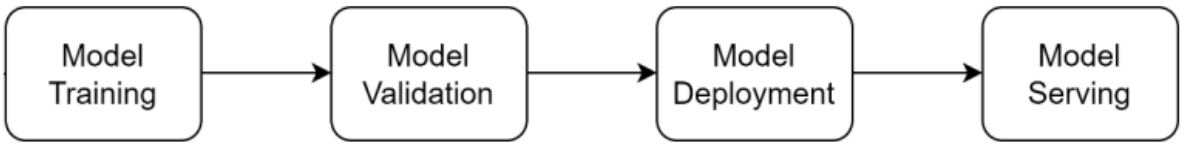

Figura 12: Diagrama de flux de treball del model

El flux de treball del model es defineix al fitxer **model-workflow-resource.yml**, dins de la carpeta de recursos. Comprèn quatre tasques seqüencials (Figura 12):

1. **Model d'entrenament:** Aquesta tasca està associada amb els fitxers de la carpeta d'entrenament i implica entrenar el model d'aprenentatge automàtic, registrant-lo a MLflow. Aquesta tasca està preparada per utilitzar opcionalment el Feature Store de Databricks per consumir característiques.

   El codi d'aquesta tasca es pot trobar al fitxer:

training\notebooks\ModelTraining.py
Cal tenir en compte que el model s'ha de registrar al registre MLflow (més informació), per a habilitar el seguiment de la versió del model en el registre MLflow.

Opcionalment, aquesta tasca avalua el consum d'energia, en kWh, i les emissions de CO_2, en kg, durant el model training, guardant aquests resultats en una taula. Es pot trobar més informació sobre els resultats a la documentació de CodeCarbon.

2. **Validació del model:** Aquesta tasca s'associa amb els fitxers de la carpeta de validació, i en aquesta etapa el model entrenat s'avalua utilitzant un subconjunt de validació de dades per validar el seu rendiment contra dades no vistes. Si les mètriques d'avaluació definides compleixen un llindar donat, la versió del model passa la fase de validació i se li assigna l'àlies de «Challenger» a MLflow.

   El codi d'aquesta tasca es pot trovar al fitxer:

   validation\ModelValidation.py

   És al fitxer validation.py on **s'han de definir mètriques i llindars personalitzats**, així com la configuració de l'avaluador. Podeu trobar més informació en el següent enllaç. No s'esperen canvis al notebook de validació respectiu, ja que el notebook només implementa el procés de validació, que es defineix per les funcions al fitxer validation.py.
   Opcionalment, aquesta tasca avalua el consum d'energia i les emissions de CO_2 durant el validation model, guardant aquests resultats en una taula.

3. **Desplegament del model (model deployment):** Aquesta tasca està associada amb els fitxers de la carpeta deployment/model_deployment . Un cop validat, el model es desplega dins del registre MLflow, assignant l'àlies "Champion" a la versió més recent, fent-lo disponible per fer prediccions en temps real.
   El desplegament de models, en la seva implementació actual, requereix canvis mínims. Això es deu al fet que la seva funció principal és simplement promoure el model dins del registre MLflow. De manera similar, la inferència per lots tampoc necessita modificacions addicionals, ja que només recupera el model amb l'àlies de 'champion' del registre i l'utilitza per generar prediccions.

   El codi d'aquesta tasca es pot trobar al fitxer:

   validation\ModelDeployment.py

4. **Model de servei:** Aquesta tasca està associada amb els fitxers de la carpeta serving/model_serving, i crea un punt final de servei (serving endpoint) per accedir al model, permetent que es generin prediccions a partir de dades d'entrada.

   El codi d'aquesta tasca es pot trobar al fitxer:

   serving\notebooks\ModelServing.py

Pot ser necessari realitzar certs ajustaments en funció de les entitats de servei o de les càrregues de treball previstes, per exemple

Aquest flux de treball no té programa (Schedule), el que significa que només s'ha d'executar quan es desplega o s'ajusta el model. Cada tasca llegeix i passa variables rellevants de les tasques anteriors i posteriors, respectivament.

3.2.4 Flux de treball Batch Inference

El fitxer de configuració batch-inference-workflow-resource.yml defineix la configuració del flux de treball per el Batch Inference, definint un nou clúster amb un worker i una configuració específica de Spark, juntament amb permisos per a l'accés de l'usuari. El clúster està etiquetat per al seguiment de FinOps amb un nom de projecte específic.

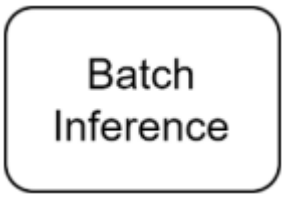

Figura 13: Diagrama del flux de treball Batch inference

La feina consisteix en una sola tasca anomenada Batch Inference, que executa un notebook per al batch inference amb el model “Champion” actual situat a:

deployment/batch_inference/notebooks/BatchInference.py.

El treball inclou permisos que permeten als usuaris veure'l, i les notificacions per correu electrònic es poden configurar per a alertes d'èxit o fallades. Processa les dades d'entrada com a features a predir pel model, i desa la sortida com una taula de Catàleg d'Unitat. Aquest flux de treball no té programa configurat, el que significa que s'ha d'executar sempre que es consideri necessari. Si el feature store està habilitat, les dades d'entrada es poden combinar amb features de taules de característiques.

Opcionalment, aquest treball avalua el consum d'energia i les emissions de CO_2 durant la inferència per lots, guardant aquests resultats en una taula.

3.2.5 Flux de treball de monitoratge

El fitxer de configuració monitoring-workflow-resource.yml defineix un treball de monitorització de model en un entorn Databricks que s'executa en un clúster de treball designat. A més, la configuració inclou permisos, programació opcional si la planificació de seguiment està activada, i notificacions per correu electrònic tant per a l'èxit com per a la fallada, facilitant el seguiment organitzat i automatitzat del model.

Garanteix un seguiment adequat de les dades i el rendiment del model, avaluant tant la deriva de les dades com les prediccions de models.

Figura 14: Diagrama del flux de treball de monitoratge

El flux de treball del monitoratge assegura un seguiment adequat de les dades i el rendiment del model, avaluant tant la deriva de dades com les mètriques de predicció de models si s'escau (és a dir, si la veritat del terreny està disponible). Aquest flux de treball comprèn tres tasques (Figura 14):

1. **CreateMonitoringTable:** Aquesta tasca crea una taula que conté dades de seguiment tant d'inferència per lots com de fonts de dades que serveixen models. S'extreuen les dades d'inferència de les taules per lots i els punts finals de servei, que després s'uneixen en una sola taula per al seguiment de prediccions i metadades com la versió del model i la marca de temps. La taula resultant es desa al format Delta per al seu ús en tasques posteriors.

   El codi d'aquesta tasca es pot trovar al fitxer:

   \monitoring\notebooks\CreateMonitoringTable.py

2. **CreateQualityMonitor:** Aquesta tasca estableix un monitoratge de qualitat per avaluar el rendiment del model, amb comparacions opcionals de dades de referència si escau. El monitoratge està configurat amb columnes de model específiques com la columna de predicció (predicció), marca horària (inference_timestamp), i versió de model. Aquestes columnes s'agreguen basant-se en granularitats especificades (per exemple, cada hora) per avaluar el rendiment del model en el temps. La inclusió opcional de les dades de referència permet al monitoratge comparar prediccions contra valors de veritat del sòl, millorant la capacitat de rastrejar la deriva del model o problemes de rendiment.

   El codi d'aquesta tasca es pot trovar al fitxer:

   \monitoring\notebooks\CreateQualityMonitor.py

3. **CreateScheduleAlerts:** Aquesta tasca automatitza les alertes de seguiment. Programa consultes periòdiques (utilitzant cron_expression) per avaluar les mètriques definides en metrics_to_monitor. Aquestes mètriques es comparen amb llindars especificats (metric_violation_thresholds). Si es produeixen violacions, s'activen les alertes i s'envien notificacions al correu electrònic definit (notification_email). Cada alerta està vinculada a una consulta per a mètriques específiques i es pot personalitzar en funció del cas d'ús i de l'operand (per exemple, més gran que, menys que).

   El codi d'aquesta tasca es pot trovar al fitxer:

   \monitoring\notebooks\CreateScheduledAlerts.py

El flux de treball es pot programar per executar-se a intervals regulars segons un horari. Si aquesta programació està habilitada, la tasca s'executarà automàticament i comprovarà qualsevol violació en les mètriques de rendiment del model. El monitoratge de qualitat també crea un quadre de comandament que es pot accedir a la pestanya Dashboard.

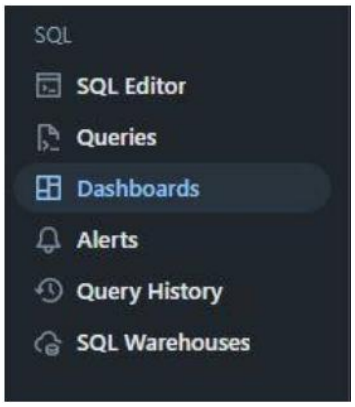

Figura 15: Accés al Dashboards al Databricks

3.3 Secrets d'Azure i Databricks

Aquest document recomana l'ús de secrets principalment per emmagatzemar de manera segura claus API, entitats de servei i credencials d'usuari. Les convencions de nomenclatura per a scopes i claus secretes són establertes i proporcionades pel proveïdor de la PTD.

Per tractar la gestió de secrets, obriu un tiquet:

- Tiquet: Remedy
- Aplicació: Plataforma Transversal de Dades de la Generalitat de Catalunya
  En desenvolupament, han de seguir la mateixa nomenclatura que els Databricks workspaces en preproducció i producció (gestionats pel proveïdor de la PTD).

Hi ha dos tipus de gestió de secrets

- **Azure Key Vault-Backed Secrets** - Permet fer referència als secrets emmagatzemats a la Azure Key Vault. Aquests àmbits proporcionen una interfície de només lectura a la Azure Key Vault, el que significa que els secrets s'han de gestionar directament a la Azure Key Vault.
- **Databricks-Backed Secrets** - Aquests àmbits emmagatzemen secrets en una base de dades xifrada gestionada per Azure Databricks. A diferència dels àmbits recolzats per Azure Key Vault, els àmbits de secrets recolzats per Databricks permeten la gestió de secrets directament dins de Databricks.

La recomanació és utilitzar Azure Key Vault-Backed Secrets, tenint en compte que el proveïdor de la PTD utilitza aquesta opció per PRE i PRO. D'aquesta manera, tots els secrets utilitzats en l'azure es poden centralitzar en Key Vault.

Per obtenir informació més detallada, consulteu Secret management - Azure Databricks | Microsoft Learn.

4. Proves

Les proves són essencials per assegurar que cada part d'un sistema funciona com s'esperava, tant pel seu compte com quan s'integra amb altres components. En un cas d'ús de MLOps, això ajuda a assegurar que el rendiment del model, les *pipeline*s de dades i la funcionalitat general són precises i fiables. Mitjançant proves regulars, es poden detectar problemes aviat, evitar errors que arribin a la producció, i es pot assegurar un alt nivell de confiança en la base de codi. S'han desenvolupat un conjunt de proves, però se'n poden afegir d'altres d'acord amb les funcions afegides als fluxos de treball.

4.1 Proves unitàries

Es tracta de proves per a components individuals o funcions aïllades. Per exemple, les proves unitàries comproven que una funció que gestiona la ingesta de dades processa els fitxers correctament o que el pas de preparació de dades d'un model elimina les columnes esperades. Les proves de la unitat se centren en petites peces de funcionalitat ben definides per assegurar que es comporten com es pretenia. Aquests es troben a la carpeta t**ests/unit_tests .**

4.2 Proves d'integració

Aquestes proves se centren en verificar que múltiples components treballen junts correctament. Per exemple, les proves d'integració poden garantir que un model entrenat es pugui servir des d'un punt final de Databricks i respongui correctament a les sol·licituds externes. L'objectiu és simular les interaccions del món real entre diferents parts del sistema. Les proves d'integració es troben a la carpeta **tests/integration_tests.**

5. Repositori Git & Pipelines CI/CD

Aquesta secció descriu els processos automatitzats per integrar i desplegar aplicacions als entorns PRE i PRO. Gestionats per l'equip de SIC+, aquestes *pipeline*s estan dissenyades per agilitzar la gestió del cicle de vida, millorar l'automatització i garantir processos de desplegament fiables.

Per obtenir més detalls sobre aquest tema, consulteu la documentació oficial CTTI:

https://canigo.ctti.gencat.cat/plataformes/ghec/

Per sol·licitar un repositori Git i les *pipeline*s CI/CD necessaris, s'ha de demanar a través de **JIRA** amb el servei de **ACOCLDSIC**, amb la següent informació:

General application information

Department code	Entity code	Maintenance lot	Application Code	Component code	Acronim name	*Environments	*Cloud
PRE	CTTI				STA	dev,pre,pro	azure

List of repositories

Technic component name	*Repository type	*Category	*Engine	*Technology & version	*Builder & version
	databricks	databricks	N/A	python	python 3.x

Figura 16: Exemple de sol·licitud d'aplicacióde CI/CD

On:

- S'ha de validar el lot de manteniment, el Codi d'Aplicació (*ApplicationCode*) i el Codi de Component (*Component Code*).

- El nom del component tècnic (*Technic component name*) ha de seguir la nomenclatura de SIC+ i PTD.
- Els altres valors haurien de mantenir-se igual, però confirmeu-ho segons el vostre cas específic.

Les subseccions següents ofereixen una breu descripció dels passos de cada canalització.

5.1 Desenvolupament (DES)

S'utomatitza el procés CI per a l'entorn de desenvolupament (branca DES). Està dissenyat per assegurar que el codi compleix els estàndards de qualitat i està lliure de vulnerabilitats de seguretat abans del desplegament. A més, aplica validació per desplegar el *bundle*. El flux de treball (*https://canigo.ctti.gencat.cat/plataformes/ghec/workflows/gh-definicio-workflows/*) hauria d'estar configurat per realitzar les següents tasques clau:

- **Validació i desplegament a Databricks DES:** Valida el projecte a l'entorn de preproducció de Databricks, simulant un estat proper a la producció.
- **Anàlisi SonarQube**: Utilitza SonarQube per analitzar el codi a la recerca d'errors, vulnerabilitats i "code smells" (indicadors de problemes en el codi). Utilitza SonarQube de CTTI o la teva pròpia instància.

5.2 Preproducció (PRE)

Aquest flux de treball és un pas crític previ a la producció, ja que permet provar el comportament del sistema en un entorn controlat similar a la producció. A més, executa les proves unitàries configurades en el codi i verifica les proves d'integració després de desplegar el *bundle*. El flux de treball hauria d'estar configurat per dur a terme les següents tasques clau:

- **Anàlisi SonarQube**: Utilitza SonarQube per analitzar el codi a la recerca d'errors, vulnerabilitats i "code smells" (indicadors de problemes en el codi). Utilitza SonarQube de CTTI o la teva pròpia instància.
- **Unit Testing**: Executa proves unitàries amb *pytest* per validar el funcionament correcte de funcions i mòduls individuals de manera aïllada.
- **Proves d'integració:** Després del desplegament, es poden executar proves d'integració per assegurar que els diferents components funcionen correctament junts en l'entorn de preproducció. Aquestes proves s'han de dissenyar i afegir segons les necessitats específiques.

5.3 Producció (PRO)

La *pipeline* de Producció (PRO) és un component crucial del procés CI/CD, dissenyat per assegurar el desplegament fluid, fiable i escalable dels models d'aprenentatge automàtic a l'entorn de producció. Aquest *pipeline* és gestionat i monitorat pels equips de SIC+ per mantenir alts estàndards de rendiment i fiabilitat.

Seguint aquests processos estructurats, les *pipeline*s CI/CD gestionen eficaçment el cicle de vida del desplegament, assegurant aplicacions d'alta qualitat, segures i amb un bon rendiment tant en entorns de preproducció com de producció.