Are Datascientists Embracing DevOps Principles for Model Versioning?

Auteurs

Nous sommes quatre étudiants en dernière année à Polytech Nice-Sophia spécialisés en Architecture Logicielle (AL) :

- *Badr AL ACHKAR*
- *Nadim BEN AISSA*
- *Sourour GAZZEH*
- *Imene YAHIAOUI*

Contexte :

La transition de DevOps à MLOps, intégrant les principes de l'intégration continue et du déploiement continu (CI/CD) aux systèmes d'apprentissage automatique, s'accompagne d'une importance croissante accordée à MLflow en tant qu'outil central dans la gestion du cycle de vie des modèles. Dans ce contexte évolutif, la question de la gestion des versions des modèles devient cruciale pour garantir une intégration harmonieuse dans les pipelines CI/CD, tout en maximisant l'efficacité du développement et du déploiement.

Gestion des versions avec MLflow :

MLflow révolutionne la gestion des modèles d'apprentissage automatique en unifiant le processus du développement à la production. La première phase de notre enquête se concentre sur la manière dont MLflow gère les versions des modèles.

- Comment MLflow stocke les modèles ? Comment sont-ils enregistrés, organisés et récupérés ?

Les résultats de cette interrogation sont disponibles dans la documentation officielle et ont été confirmés par plusieurs articles scientifiques, dont les références sont fournies dans la bibliographie.

MLflow enregistre les modèles sous forme d'artefacts, utilisant divers formats appelés "flavors", qui correspondent à différents formats spécifiques de bibliothèques de machine learning, tels que TensorFlow, PyTorch, Scikit-Learn. Ces modèles sont organisés dans un "registre" structuré, chacun possédant un nom unique, des versions, des étapes de transition (telles que développement, production, archivage) et d'autres métadonnées.

La gestion des versions en Mlflow est automatisée, où chaque modèle enregistré peut avoir une ou plusieurs versions. Lorsqu'un nouveau modèle est ajouté au registre des modèles, il est enregistré en tant que version 1. Chaque nouvel enregistrement sous le même nom incrémente automatiquement le numéro de version.

Il est possible d'adapter manuellement cette gestion de versions à travers l'utilisation d'alias, de tags et de descriptions :

- Les alias permettent de pointer vers une version spécifique du modèle, facilitant ainsi le référencement via l'URI models:/@. Ils offrent une référence mutable et nommée à des versions spécifiques, simplifiant leur déploiement.
- Les tags permettent de catégoriser les modèles selon leur fonction ou statut.
- Les annotations et descriptions en Markdown fournissent des informations cruciales telles que des descriptions de version, des ensembles de données utilisés, etc.

Exploration du versionnement des modèles sur Hugging Face :

La deuxième phase de notre étude se tourne vers l'exploration des modèles sur Hugging Face, en réponse à l'absence de projets MLflow dédiés à l'analyse. Cette absence peut s'expliquer par le fait que MLflow est principalement adopté par des entreprises, dont beaucoup ne rendent pas publics leurs modèles. En contraste, Hugging Face offre une plateforme qui propose des modèles développés par d'importantes entreprises telles que Facebook et Google.

1. ***Est ce que les modèles publiés sur Hugging Face adoptent un schéma de versionnement traditionnel ou part plutôt sur le versionnement libre ?***
2. ***Le type de tâches (image-to-text, reinforcement, etc.) influence-t-il le schéma de versionnement ?***
3. ***Existe-t-il des tendances de versionnement spécifiques dans les grandes entreprises utilisant Hugging Face?***

Hypothèses & Expériences et Analyse des résultats

1. ***Adoption du Versionnement Traditionnel vs Versionnement Libre des Modèles***

a. ***Hypothèse : Le versionnement des modèles ne suit pas une approche traditionnel - (major,minor,patch)***

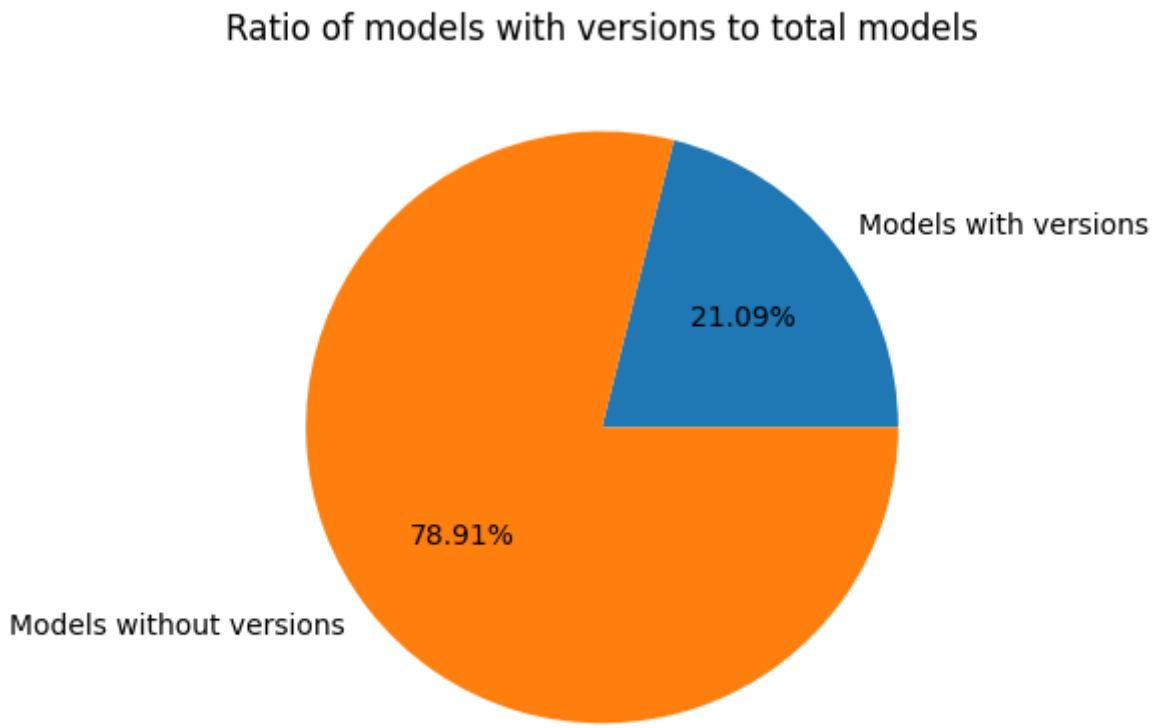
L'observation des modèles sur Hugging Face suggère que la numérotation des versions des modèles ne suit pas un motif prévisible, contrairement à la structure classique de versionnement. Il semble y avoir une variation dans la manière dont les versions sont étiquetées. Cette variation dans l'étiquetage des versions suscite la question suivante : Est-ce que les modèles publiés sur Hugging Face adoptent un schéma de versionnement traditionnel ou part plutôt sur le versionnement libre ?

b. ***Expérience : Extraction d'un échantillon important des modèles versionnées et poussé sur Hugging Face, et calcule de l'adoption du versionnement traditionnel***

Dans une première étape, nous avons utilisé une approche de web scraping plutôt naïve pour extraire les noms des modèles. Cette méthode a été mise en œuvre à l'aide de la bibliothèque Python BeautifulSoup, laquelle est conçue pour effectuer l'analyse syntaxique de documents HTML et XML. Nous avons ensuite calculé le ratio des modèles qui contiennent un format de versionning sémantique (vX, vX.X, vX-X, vX.X.X, etc).

c. ***Résultat :***

Cette premiere expérience nous a permis de confirmer notre hypothèse, avec un ratio de 79% des modèles qui ne suivent pas le versionnement traditionnel.



2. ***Influence de la tâche du modèle sur son versionnement***

    a. ***Hypothèse : La nature de la tâche du modèle influence la > façon avec laquelle on le versionne***

    b. ***Expérience : Analyse des versionnement par tâche et > extraction des patterns réccurents***

    c. ***Résultat :***

Des tâches comme reinforcement learning adopte le versionnement traditionnel (constitue la majorité des 21% qui reste)

Les autres ont des patterns différents, il n'y a pas un consensus global

3. ***Tendances de Versionnement dans les Grandes Entreprises***

    a. ***Hypothèse : Les grandes entreprises standardise leurs manière > de versionner les modèles ML en interne***

    b. ***Expérience : Analyse des modèles ceux publié par Google, > OpenAI et Meta, individuellement, et extraction des patterns > adoptée.***

    c. ***Résultat :***

Il n'y a pas un pattern de versionnement global par entreprise non plus

Conclusion

- nom des versions => nom de branches?

Références