École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS,
FRANCE
(33)2-47-36-14-14
www.polytech.univ-tours.fr

Projet libre

Guide d'MLOps

Étudiant
Théo BOISSEAU

Encadrant
Nicolas RAGOT

Table des matières

I. Introduction

A. Qu'est-ce que l'MLOps ?

MLOps (Machine Learning Operations) est une pratique qui unit développement logiciel et production de modèles ML. Il s'agit d'une approche collaborative entre équipes de développement, données, DevOps et métiers pour garantir fiabilité, évolutivité et déploiement facile en production.

MLOps comprend plusieurs éléments clés, tels que:

- Automatisation: Automatiser les processus de construction, de test et de déploiement de modèles pour réduire les erreurs humaines et accélérer les cycles de livraison.
- Collaboration: Faire en sorte que les équipes travaillent ensemble pour garantir la qualité des données, des modèles et des déploiements.
- Monitoring: Surveiller les modèles en production pour détecter les anomalies et les erreurs potentielles.
- Evolutivité: Faciliter la mise à jour et l'amélioration des modèles en production.

L'objectif principal de MLOps est de permettre une livraison rapide et fiable de modèles de Machine Learning en production, ce qui permet d'accélérer la découverte de valeur

B. Importance de l'MLOps pour la production de modèles de Machine Learning

MLOps est crucial pour la production de modèles ML en raison de leur complexité et de la nécessité de garantir qualité, fiabilité et évolutivité. Les entreprises peuvent surmonter les défis suivants:

- Complexité en développement de modèles: Les modèles ML peuvent être complexes. MLOps aide à automatiser et standardiser les processus pour une meilleure collaboration entre équipes de développement de logiciels, de données et de métiers.
- Qualité des données: La qualité des données est cruciale pour les modèles ML. MLOps surveille la qualité des données pour garantir que les modèles sont formés sur des données fiables.
- Fiabilité en production: Les modèles peuvent subir des charges de travail variables en production. MLOps surveille les modèles en production pour détecter rapidement les erreurs.
- Evolutivité des modèles: Les modèles peuvent être améliorés au fil du temps. MLOps facilite la mise à jour et l'amélioration des modèles en production pour répondre aux nouveaux besoins métiers.

II. Comprendre les enjeux du MLOps

A. Qualité de modèles

Pour l'évaluation de la qualité des modèles, on peut utiliser différentes métriques et techniques comme :

- Précision : mesure la capacité d'un modèle à prédire correctement. On peut la mesurer avec accuracy, sensibilité, et spécificité.
- AUC-ROC : (Area Under the Receiver Operating Characteristic Curve) mesure la capacité à distinguer les classes positives et négatives.

- Validation croisée : évalue la performance avec un ensemble de données différent de celui utilisé pour l'entraîner.
- Courbe de loss : montre la différence entre les prédictions et les valeurs réelles pour un ensemble d'entraînement/validation.

Il est aussi important de surveiller la performance en production via la disponibilité, la latence, et la fiabilité.

Pour vérifier la robustesse des modèles, utiliser la validation croisée, sur- ou sous-échantillonnée.

Pour garantir la qualité, il faut implémenter des processus rigoureux de revue et validation interne avant déploiement en production.

B. Collaboration et traçabilité, Automatisation et intégration continue

L'automatisation et l'intégration continue sont importants pour produire des modèles de Machine Learning fiables et efficaces. Automatiser le processus d'entraînement, d'évaluation et déploiement des modèles améliore la qualité et la vitesse de développement. Une infrastructure pour l'intégration continue, permettant de détecter automatiquement les erreurs, garantit la stabilité et la qualité des modèles déployés.

La collaboration et la traçabilité sont aussi importantes. Il peut inclure une documentation détaillée, gestion des versions, tests unitaires et d'intégration, surveillance des performances en production. Certaines pratiques pour facilitent cela incluent :

- Plateformes de gestion de modèles, comme TFX et MLflow qui centralisent les modèles, pipelines et données pour améliorer la collaboration et la traçabilité.
- Documentation complète des modèles, pipelines et données, qui aide à comprendre les algorithmes, métriques, paramètres utilisés.
- Standardisation des processus de formation et déploiement, qui facilite la collaboration et la traçabilité.
- Systèmes de pipeline d'intégration continue et déploiement, comme Jenkins, CircleCI, TravisCI, qui automatisent les pipelines et améliorent la traçabilité.

Il est crucial de travailler en étroite collaboration avec les équipes d'infrastructure pour déployer des modèles fiables, évolutifs et élastiques. Cela inclut une surveillance en production pour détecter rapidement les anomalies et problèmes. Des outils populaires pour l'automatisation et l'intégration continue incluent Jenkins, CircleCI, TravisCI et GitLab CI/CD, ainsi que des solutions ML spécifiques telles que TFX et MLflow.

III. Établissement de la stratégie MLOps

A. Évaluation des besoins en matière de production de modèles

Evaluer les besoins en production de modèles est crucial pour implémenter une stratégie MLOps efficace. Il s'agit de comprendre les critères de qualité, les délais pour déployer des modèles, et l'intégration avec les systèmes existants.

- Qualité des modèles: déterminer les critères de qualité tels que la précision, la robustesse, la fiabilité et la scalabilité.
- Temps de déploiement: comprendre les délais pour déployer et faire évoluer les modèles.
- Intégration: comprendre les systèmes existants et déterminer comment les modèles s'intégreront à eux.

Evaluer ces besoins avec des enquêtes, des entretiens et des analyses de données permettra de définir les objectifs MLOps et les outils/processus nécessaires pour les atteindre.

B. Définition des processus et des workflows MLOps

Les processus et workflows MLOps décrivent les étapes nécessaires pour gérer le cycle de vie complet d'un modèle ML, depuis la conception à la maintenance en production.

1. Formation et test des modèles: Ce processus implique la sélection des algorithmes, la préparation des données et l'entraînement/test des modèles pour évaluer leur qualité. Des outils tels que TensorFlow, PyTorch, scikit-learn peuvent être utilisés.
2. Gestion des modèles: Cela implique la gestion des versions des modèles, le stockage des modèles et la documentation des processus de formation pour la traçabilité et la collaboration. Des outils comme MLflow, Datmo et Git peuvent être utilisés.
3. Mise en production: Cela implique la mise en place des modèles sur un serveur de production, leur intégration avec d'autres systèmes et la mise en place de mécanismes de surveillance pour garantir la qualité des modèles en production. Des outils tels que TensorRT, TensorFlow Serving et Apache Kafka peuvent être utilisés.
4. Surveillance continue: Cela implique la surveillance constante des modèles en production pour évaluer leur performance et déterminer s'ils nécessitent des mises à jour. Des outils tels que TensorBoard, Datadog et Prometheus peuvent être utilisés.

Il est important de documenter les processus et workflows MLOps clairement pour assurer une collaboration efficace entre les équipes impliquées. Ces processus peuvent être documentés dans des outils de gestion de projet comme Jira ou Asana ou dans des outils de gestion de documentation comme Confluence ou ReadMe.

C. Établissement d'une infrastructure MLOps

L'établissement d'une infrastructure MLOps nécessite des outils pour gérer les processus définis précédemment. Cela inclut :

1. Environnement de formation : pour former les modèles. Outils incluent TensorFlow et PyTorch.
2. Environnement de test : pour tester les modèles. Outils incluent scikit-learn et TensorFlow/PyTorch.
3. Environnement de production : pour exécuter les modèles en production. Outils incluent TensorFlow Serving et Google Cloud ML/Amazon SageMaker.
4. Surveillance : pour assurer la qualité des modèles en production. Outils incluent TensorFlow On-Boarding et Amazon SageMaker.

Il est important de noter que l'établissement d'une infrastructure MLOps efficace peut être un processus complexe et dépend fortement des besoins spécifiques de l'organisation. Il est recommandé de travailler avec des experts en MLOps et en infrastructure pour déterminer les meilleures solutions pour votre organisation.

IV. Développement de modèles

A. Entraînement et évaluation de modèles

L'entraînement consiste à ajuster les paramètres d'un modèle pour prédire avec précision grâce aux données. L'évaluation mesure la qualité des prédictions.

1. Entraînement: Choix des algorithmes, préparation des données, formation des modèles et sélection des hyperparamètres optimaux. Outils utiles incluent scikit-learn, TensorFlow et Google Cloud ML/Amazon SageMaker.
2. Évaluation: Mesure de la qualité des prédictions avec des métriques comme l'exactitude, recall et précision. Il faut utiliser des données de test différentes pour éviter l'overfitting. Outils utiles incluent les métriques de performance de scikit-learn et les fonctions d'évaluation de TensorFlow.

Il est important de faire l'entraînement et évaluation de manière cohérente et standardisée pour garantir la qualité et fiabilité des modèles en production. Les équipes MLOps peuvent utiliser des outils d'automatisation pour un processus fiable et reproductible.

B. Gestion des versions et traçabilité

La gestion des versions et la traçabilité permettent de comprendre l'évolution des modèles et résoudre les problèmes.

1. Gestion des versions: suivre les différentes versions d'un modèle de Machine Learning, y compris les informations sur les algorithmes, données d'entraînement et hyperparamètres. Outils: Git, Bitbucket, GitHub.
2. Traçabilité: suivre les étapes du cycle de vie d'un modèle, y compris entraînement, évaluation, production. Peut inclure données utilisées, métriques de performance & améliorations. Outils: Kubeflow, MLflow.

Avec ces systèmes, les équipes MLOps peuvent retracer les changements et comprendre les différences entre les versions.

C. Tests unitaires et intégration

Tests unitaires et intégration sont importants pour assurer qualité et fiabilité des modèles de ML.

1. Tests unitaires: Vérifier chaque partie du modèle séparément pour s'assurer qu'ils fonctionnent bien. Incluent des tests pour les données d'entraînement, algorithmes et résultats.

2. Tests d'intégration: Combiner les parties du modèle pour qu'ils fonctionnent ensemble. Incluent des tests pour s'assurer de l'intégration avec d'autres systèmes tels que bases de données, pipelines, etc.

Outils utiles: Bibliothèques de tests Python (unittest, pytest), outils CI (Jenkins, TravisCI).

Tests unitaires et intégration assurent que les modèles fonctionnent correctement avant déploiement en prod. Aident à détecter erreurs plus rapidement et prévenir problèmes en prod.

V. Déploiement de modèles

A. Packaging et déploiement

Packaging et déploiement sont les étapes finales dans le développement d'un modèle ML.

- Packaging: Le packaging consiste à préparer un modèle ML pour le déploiement en production. Cela inclut l'empaquetage de tous les fichiers requis en un format portable, comme les algorithmes, les données d'entrainement, les bibliothèques, etc. Docker containers et fichiers Python sont des formats courants pour le packaging de modèles ML.
- Déploiement: Le déploiement met en place un modèle ML en production, soit sur des infrastructures cloud (AWS, GCP, Azure) ou sur site. On peut également déployer les modèles ML sur des plateformes spécialisées (TensorFlow Serving, Amazon SageMaker).

Des outils tels que Docker, Kubernetes et TensorFlow Serving/Amazon SageMaker aident à effectuer le packaging et le déploiement.

En utilisant des méthodes fiables pour le packaging et le déploiement, les équipes MLOps peuvent déployer rapidement des modèles ML en production avec une réduction d'erreurs et délais. Cela contribue également à maintenir la qualité et la fiabilité des modèles ML sur le long terme, crucial pour garantir leur pertinence et efficacité.

B. Surveillance et maintenance

La surveillance et la maintenance permettent de garantir la fiabilité des modèles de Machine Learning en production.

- Surveillance: C'est la surveillance constante des modèles déployés pour détecter les problèmes. Cela inclut la performance, les prévisions, l'utilisation des ressources et la conformité réglementaire. Des outils tels que TensorBoard et New Relic sont utilisés pour surveiller.
- Maintenance: C'est la maintenance constante des modèles déployés pour les maintenir en bon état. Cela inclut la mise à jour régulière des algorithmes, des données d'entraînement, la correction de bugs et l'adaptation aux données en constante évolution.

En utilisant une surveillance et une maintenance adéquates, les équipes MLOps peuvent garantir la qualité et la fiabilité des modèles déployés en production.

En résumé, la surveillance et la maintenance sont importants pour le MLOps et doivent être intégrés dans le cycle de vie des modèles pour garantir leur qualité et fiabilité en production.

VI. Outils et technologies

A. Plateformes d'intégration continue et automatisation

Ils permettent de construire, tester et déployer des applications automatiquement. Ils sont souvent utilisés pour automatiser le processus d'intégration continue et de déploiement, ce qui aide à garantir la qualité du modèle.

- **Jenkins** : une solution très populaire et polyvalente, offrant de nombreuses fonctionnalités de CI/CD personnalisables. Cependant, il peut être difficile à installer et à utiliser pour les débutants.
- **CircleCI** : option facile à utiliser pour les équipes de développement, offrant une intégration simple avec les outils populaires tels que GitHub et AWS.
- **TravisCI** : solution populaire pour les projets open source, avec une intégration facile avec GitHub et une configuration simple.
- **GitLab CI/CD** : option intégrée pour les équipes utilisant GitLab, offrant une intégration complète avec le pipeline de version et une grande flexibilité pour les pipelines personnalisés.

B. Outils de gestion des versions

Ils permettent de gérer les différentes versions du code et du modèle, ce qui facilite la collaboration et la traçabilité. Ils sont souvent utilisés pour versionner le code et les modèles, ce qui peut aider à comprendre les modifications apportées au fil du temps.

- **Git** : très populaire qui offre une grande flexibilité et une forte communauté d'utilisateurs. Il est particulièrement adapté aux projets distribués et aux grandes équipes de développement.
- **SVN** : plus traditionnel qui convient aux projets de taille moyenne avec une équipe de développement stable. Il offre une plus grande centralisation et une meilleure gestion des autorisations que Git.
- **Mercurial** : similaire à Git, mais avec une plus grande flexibilité pour les opérations distribuées. Il est adapté aux petites équipes de développement et convient bien aux projets distribués.

C. Outils de déploiement

Ils permettent de déployer des applications en production. Ils sont souvent utilisés pour déployer des modèles de Machine Learning sur des infrastructures cloud.

- **Docker** : solution populaire pour le déploiement de conteneurs, offrant une simplicité d'utilisation pour les développeurs et une intégration facile avec de nombreuses autres technologies.
- **Kubernetes** : solution plus avancée pour l'orchestration de conteneurs, offrant une grande scalabilité, une tolérance aux pannes et une grande flexibilité pour les déploiements complexes.

- **OpenShift** : plateforme de conteneurs construite sur Kubernetes, offrant des fonctionnalités supplémentaires telles que le déploiement continu, la gestion de la sécurité et la gestion des applications dans le cloud.

D. Outils de surveillance et maintenance

Ils permettent de surveiller et de maintenir les applications en production. Ils sont souvent utilisés pour surveiller les performances et les métriques des modèles en production.

- **Grafana** : solution de visualisation de données puissante qui permet de surveiller et d'analyser les performances des applications et des infrastructures. Il offre de nombreuses options de personnalisation et de collaboration pour les équipes d'exploitation.
- **Prometheus** : solution de surveillance centrée sur les métriques, offrant une collecte de métriques rapide et fiable, ainsi qu'une grande flexibilité pour la personnalisation des alertes.
- **Nagios** : solution de surveillance traditionnelle, offrant une vaste gamme de fonctionnalités pour la surveillance des systèmes et des réseaux. Il est également facile à utiliser pour les utilisateurs débutants.

E. Outils de packaging

Ils permettent de paqueter des applications et des dépendances pour le déploiement. Ils sont souvent utilisés pour paqueter des modèles de Machine Learning et leur environnement de développement pour le déploiement.

- **Pip** : gestionnaire de paquets standard pour Python et est très populaire pour les projets simples. Il est facile à utiliser et fournit un accès à un grand nombre de paquets Python.
- **Conda** : gestionnaire de paquets plus avancé qui offre une meilleure gestion des dépendances et des environnements pour les projets plus complexes. Il est également capable de gérer des paquets pour d'autres langages de programmation, ce qui peut être utile pour les projets inter-langues.

F. Outils de pipeline de modèles

Ils permettent de construire des pipelines pour le développement et la production de modèles de Machine Learning. Ils sont souvent utilisés pour automatiser le processus de développement de modèles de Machine Learning, de leur évaluation et de leur déploiement.

- **TensorFlow Extended (TFX)** : framework développé par Google pour la production de modèles. Il est particulièrement utile pour les équipes qui utilisent déjà TensorFlow pour leur développement de modèles et qui cherchent une solution pour la production.
- **Kubeflow** : framework open source pour la mise en production de modèles sur les environnements Kubernetes. Il est particulièrement utile pour les équipes qui cherchent une solution qui prend en charge une infrastructure distribuée.

- **Apache Beam** : framework open source pour les pipelines de traitement de données distribuées. Il peut être utilisé pour la production de modèles, mais il est plus adapté aux pipelines de traitement de données traditionnels.

G. Outils de collaboration

Ils permettent de collaborer avec d'autres développeurs sur le code et les modèles.

- **GitHub** : plus grand référentiel de code open source au monde, offrant une grande communauté de développeurs et une grande variété de plugins et d'outils pour améliorer la collaboration. Cependant, les coûts peuvent augmenter rapidement pour les équipes professionnelles.
- **GitLab** : offre une intégration complète avec les pipelines de version et les pipelines de déploiement, ainsi qu'une grande flexibilité pour les pipelines personnalisés. Cependant, leur offre gratuite peut être limitée en fonctionnalités.
- **Bitbucket** : option pour les équipes plus petites, offrant un nombre illimité de référentiels privés pour les équipes de moins de cinq utilisateurs. Bitbucket propose également une intégration avec les outils de développement tels que Jira et Trello.

H. Outils de surveillance et maintenance de l'infrastructure MLOps

Ils sont importants pour surveiller les performances du système d'infrastructure MLOps et pour détecter les erreurs et les anomalies en temps réel. Cela permet une maintenance proactive pour minimiser les perturbations et les temps d'arrêt. Les équipes MLOps peuvent également utiliser ces outils pour surveiller les performances des modèles en production et pour faire des mises à jour en fonction des données en temps réel.

- **Datadog** : option populaire de taille moyenne à grande, offrant une interface utilisateur facile à utiliser et des fonctionnalités de surveillance avancées.
- **Nagios** : solution mature pour la surveillance de la performance, mais peut être complexe à configurer et à utiliser pour les débutants.
- **Zabbix** : option éprouvée de grande taille, offrant une intégration approfondie avec les infrastructures complexes et un large éventail de fonctionnalités de surveillance.
- **Grafana** : option populaire pour la visualisation de données de performance, offrant une intégration avec de nombreuses sources de données et une flexibilité pour les tableaux de bord personnalisés.
- **InfluxDB** : base de données de séries chronologiques pour la surveillance et la surveillance des performances, offrant une intégration avec Grafana et une gestion efficace des données de performance à grande échelle.
- **Prometheus** : solution de surveillance de performance open source, populaire pour sa flexibilité et sa scalabilité pour les environnements de production complexes.

I. Outils de packaging et de déploiement de modèles

Ils permettent de faire le packaging de modèles de Machine Learning pour une distribution et un déploiement en production en toute sécurité. Ils facilitent également la gestion des dépendances, la scalabilité et la résolution des erreurs. Les équipes MLOps peuvent utiliser ces outils pour déployer les modèles en production en toute sécurité et de manière efficace.

- **Docker** : solution de conteneurisation populaire, qui permet de déployer et de faire tourner des applications de manière consistante sur toutes les plateformes.
- **Kubernetes** : système de gestion de cluster pour les conteneurs, qui offre une grande scalabilité et une gestion centralisée des applications.
- **Helm** : gestionnaire de package pour Kubernetes, qui facilite le déploiement et la gestion d'applications dans le cloud.
- **Ansible** : outil de configuration et de gestion de déploiement pour les applications, qui offre une grande flexibilité pour les tâches de configuration et de déploiement.
- **AWS Elastic Beanstalk** : plateforme de déploiement automatisée pour les applications web sur AWS, qui offre une gestion facile des ressources et une intégration avec les autres services AWS.

J. Technologies d'infrastructure cloud

Ces technologies permettent aux équipes de déployer en production des systèmes MLOps tout en garantissant une haute disponibilité, une évolutivité en temps réel et une sécurité accrue pour les données et les modèles.

- **AWS (Amazon Web Services)** : offre une variété de services, est facile à utiliser et est flexible.
- **Google Cloud Platform** : se spécialise en IA et ML, est facile à utiliser et est flexible.
- **Microsoft Azure** : se concentre sur la gestion de données et la mobilité, et peut être complexe pour les débutants.
- **IBM Cloud** : mise sur les technologies blockchain et Watson.

Les coûts varient considérablement entre les différents fournisseurs, il faut donc comprendre les modèles de tarification et les options de facturation. La conformité et la sécurité sont des préoccupations majeures. Il est donc important de vérifier les certifications et les protocoles de sécurité pour garantir la protection des données et des applications. Le support et la documentation peuvent également influencer le choix selon les besoins.