MLOps: життя моделі після тренування

В програмі

1. Machine Learning Operations та що потрібно врахувати після готовності моделі.

2. Можливості деплойменту моделі.

3. Бібліотека streamlit для простого деплойменту і демонстрації роботи моделі.

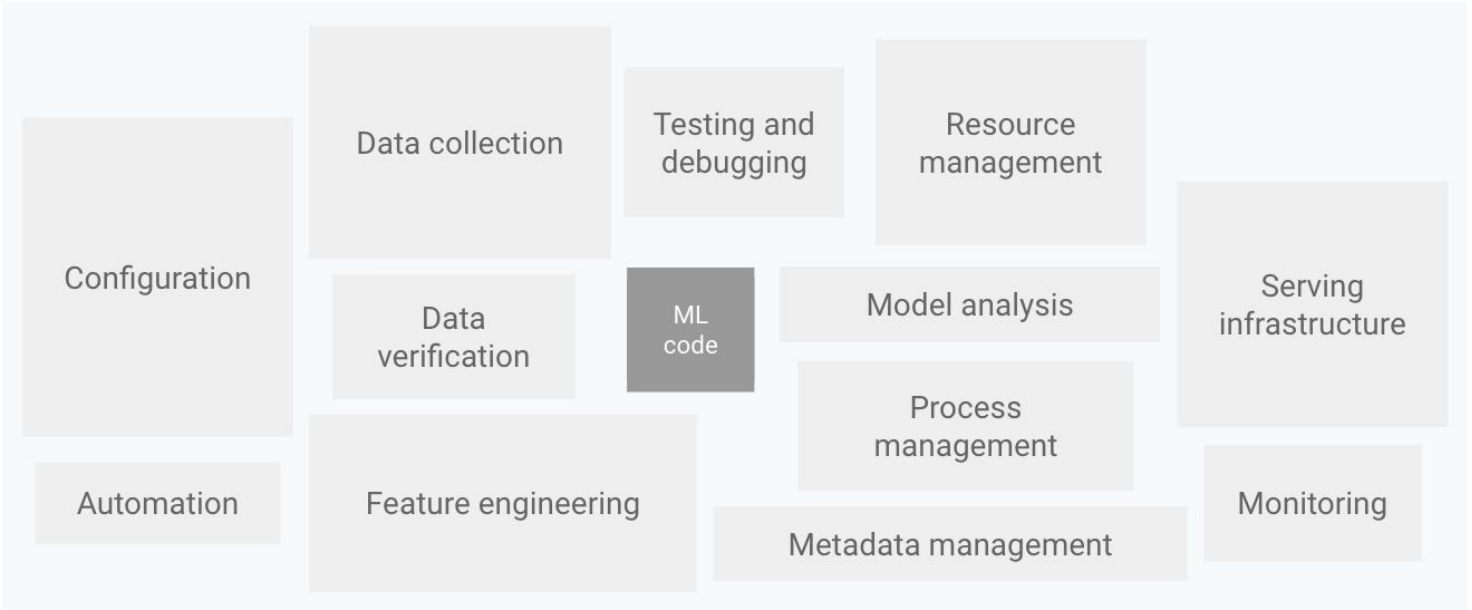
Що таке MLOps?

MLOps розшифровується як Machine Learning Operations і означає процес управління життєвим циклом машинного навчання, від розробки до розгортання і моніторингу. Він включає такі завдання, як

- **Відстеження експериментів/Experiments tracking:** Відстеження експериментів і результатів для визначення найкращих моделей
- **Розгортання моделей/Mode deployment:** Розгортання моделей у виробництво і надання їх додаткам
- **Моніторинг моделей/Model monitoring:** Моніторинг моделей для виявлення будь-яких проблем або погіршення продуктивності
- **Перенавчання моделей/ Model retraining:** Перенавчання моделей на нових даних для покращення їхньої продуктивності

MLOps має важливе значення для забезпечення надійності, масштабованості та підтримки моделей машинного навчання у виробничих середовищах.

Сучасну ML систему часто зображають такою діаграмою



https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

MLOps - величезна тема

І моя задача зараз - показати, що після тренування моделі ще є багато роботи, і якою саме вона може бути, аби ви це чітко розуміли і могли впроваджувати в своїх проєктах.

Що потрібно врахувати після готовності (натренованості) моделі

1. Розгортання моделі: Після завершення тренування модель потрібно розгорнути, щоб зробити її доступною для використання іншими системами або користувачами. Часто це робиться у вигляді веб-сервісу (наприклад, REST API або gRPC).

Середовища для деплойменту

Для забезпечення стабільності та гнучкості деплоймент зазвичай відбувається на кілька середовищ:

- **Розробницьке середовище (dev):** На цьому етапі модель тестується розробниками для перевірки працездатності, коректності передбачень, а також для виявлення можливих помилок і недоліків. Dev-середовище також використовується для A/B-тестування різних версій моделей.
- **Тестове середовище (test/staging):** Це середовище максимально наближене до продакшн, але використовується для більш масштабного тестування з реальними даними. Воно дозволяє переконатися, що нова версія моделі коректно працює і не вплине негативно на існуючі процеси в продакшн.
- **Продакшен (prod):** Це середовище, де модель працює з реальними користувачами та даними. Деплоймент на продакшен має бути безпечним, стабільним і контрольованим, оскільки будь-яка помилка може призвести до значних втрат.

Розгортання на кілька середовищ допомагає мінімізувати ризики помилок і деградації продуктивності. Після деплою на тестове або dev середовище, можна поступово впроваджувати модель у продакшн, наприклад, за допомогою стратегії **blue-green deployment** (два ідентичні середовища: одне старе, одне нове) або **canary release** (поступове введення моделі для невеликого сегменту користувачів).

Більше про різні типи деплойменту: https://www.harness.io/blog/blue-green-canary-deployment-strategies

Що ще потрібно врахувати після готовності (натренованості) моделі

2. **Моніторинг продуктивності моделі:** Після деплою моделі в продакшн важливо постійно відстежувати її продуктивність. Це включає моніторинг

- точності передбачень,
- помилок,
- часу відповіді та використання ресурсів (процесора, пам'яті).

Модель може деградувати з часом через *зміни у вхідних даних* або в *бізнес-середовищі* (концептуальний дрейф).

Крім того, важливо налаштувати *автоматизовані оповіщення* для виявлення аномалій у продуктивності моделі, таких як раптові стрибки в кількості помилок або збільшення часу відповіді. Це дозволяє вчасно вживати заходів щодо коригування або оновлення моделі.

Що ще потрібно врахувати після готовності (натренованості) моделі

3. **Версіонування моделі:** Необхідно підтримувати версіонування моделі та всіх артефактів, пов’язаних з нею (дані, код, параметри тощо).

Це дозволяє відстежувати зміни, експериментувати та легко відкотитися до попередньої версії, якщо нова версія виявиться неефективною. Можна використовувати системи керування версіями, такі як DVC (Data Version Control) для даних та моделей, або MLflow для відстеження експериментів та управління моделями.

Що ще потрібно врахувати після готовності (натренованості) моделі

4. **Ретренування моделі:** Планування процесу ретренування моделі, коли її продуктивність падає або коли з'являються нові дані. Це може бути автоматизовано за допомогою тригерів або за фіксованим розкладом. Наприклад, можна налаштувати ретренування, коли точність моделі падає нижче певного порогу, або щомісяця для адаптації до нових даних.

Додаткові аспекти MLOps для ефективного деплою

- **Логування та аудит:** Всі виклики до моделі та її відповіді повинні логуватися для подальшого аналізу. Логи також допомагають у налагодженні та відстеженні поведінки моделі в реальному часі.
- **Тестування та валідація моделей:** Перед деплоєм важливо провести автоматизовані та ручні тести, які включають валідацію моделі на свіжих даних, тестування навантаження (load testing), та тестування стабільності. Це допоможе переконатися, що модель готова до роботи в умовах реального навантаження.
- **Забезпечення відмовостійкості:** Забезпечити механізми відкату (rollback) та автоматичного перемикання на попередню версію моделі в разі виявлення критичних помилок або збоїв у роботі нової версії.

Як можна задеплоїти ML модель?

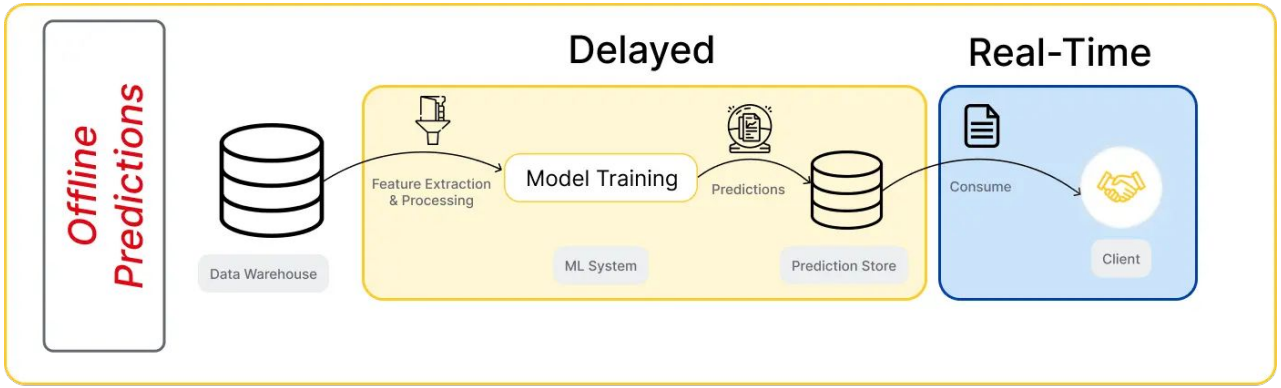
Опції деплойменту моделей

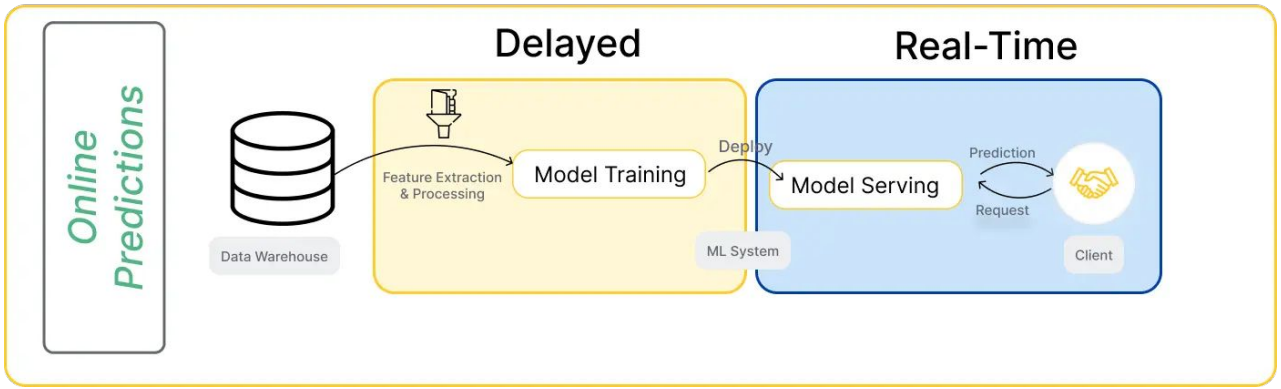
Вибір правильної опції деплойменту залежить від конкретних потреб і обмежень проєкту, таких як швидкість відповіді сервісу, масштабованість, безпека та вартість.

Існує кілька популярних підходів до делойменту моделі, від простих API-сервісів до складних архітектур з використанням хмарних сервісів, контейнеризації та безсерверних обчислень.

Розуміння цих опцій допоможе вам прийняти оптимальне рішення для успішного розгортання ваших ML-моделей.

Online vs Offline Predictions



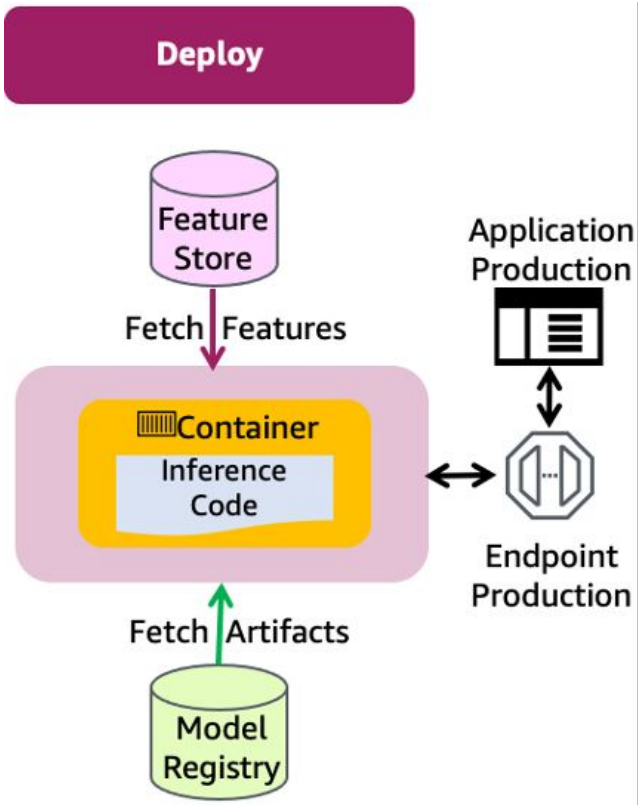


REST API або gRPC сервіс

Що це: Модель розгортається як веб-сервіс (наприклад, REST API або gRPC), що дозволяє робити HTTP-запити до сервера для отримання передбачень. Це один з найпоширеніших способів інтеграції моделей машинного навчання в існуючі програми або системи.

- **Переваги:**
  - Простота інтеграції з будь-якою мовою програмування, яка підтримує HTTP-запити.
  - Можливість швидко масштабувати за допомогою контейнеризації (Docker) та оркестрації (Kubernetes).
  - Підходить для онлайн-обробки (режим реального часу) запитів.
- **Недоліки:**
  - Може потребувати додаткової оптимізації для роботи з великим навантаженням.
  - Вимагає певної інфраструктури для забезпечення високої доступності та безпеки.

Найпопулярніші фреймворки для ML Model API - Flask, FastAPI. Приклад деплойменту з FastAPI.



Часто деплоймент буде виглядати так

Використання сервісів хмарних платформ (AWS SageMaker, Google AI Platform, Azure ML)

Що це: Використання спеціалізованих сервісів для деплойменту ML-моделей, які надають готові рішення для розгортання, моніторингу та обслуговування моделей.

Переваги:

- Легкість у використанні та швидке розгортання.
- Підтримка автоматизованого масштабування та балансування навантаження.
- Інтеграція з іншими хмарними сервісами (наприклад, базами даних, сервісами аналітики).
- Можливість використання вбудованих інструментів для моніторингу продуктивності та ретренування.

Недоліки:

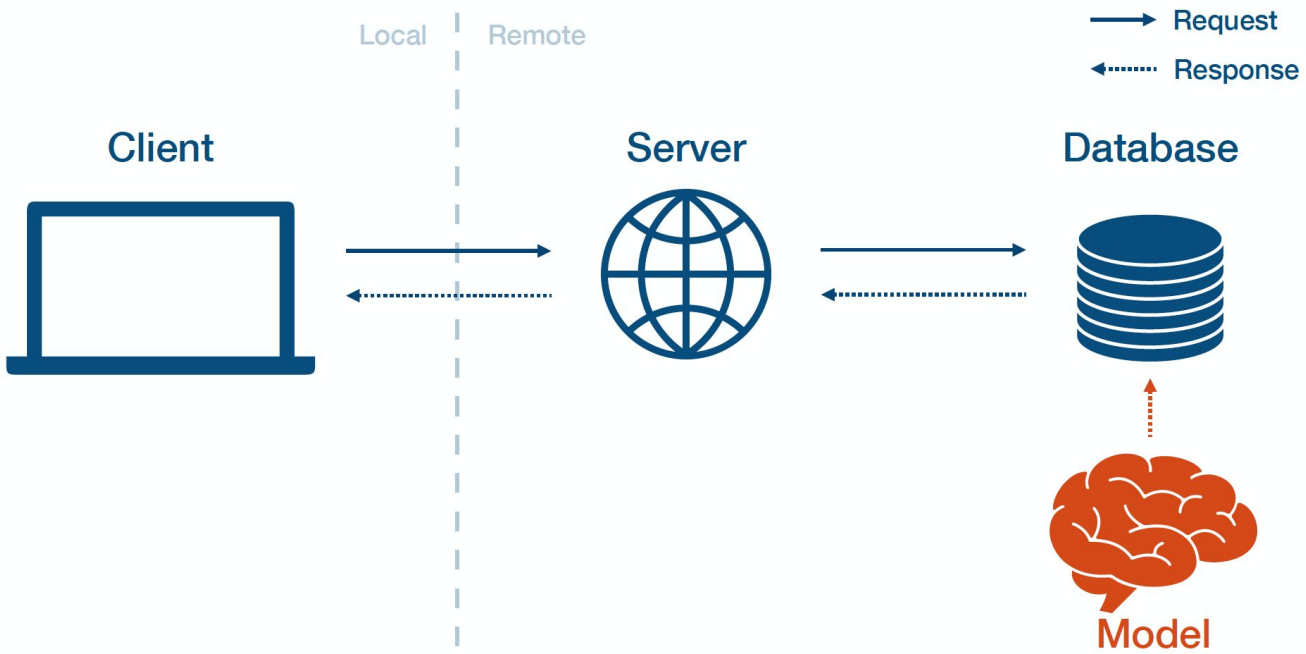
- Залежність від конкретного хмарного провайдера (vendor lock-in).
- Може бути дорожче, особливо для великих обсягів даних або моделей з високими вимогами до ресурсів.

Приклад розгортання з AWS Sagemaker.

Batch Processing

- **Що це:** Використання моделі для пакетної обробки даних (batch processing). Це означає, що передбачення робляться не в режимі реального часу, а періодично на великому обсязі даних.
- **Переваги:**
  - Підходить для обробки великих обсягів даних, коли реальний час не є критичним.
  - Можливість використання великих обчислювальних потужностей для швидкої обробки.
  - Оптимізація ресурсів (можливість запускати на дешевших ресурсах або у неробочі години).
- **Недоліки:**
  - Затримка в отриманні результатів, оскільки обробка проводиться періодично.
  - Не підходить для застосунків, що вимагають миттєвих передбачень (наприклад, чат-боти, системи рекомендацій в реальному часі).

Batch Processing



Streamlit: web application для вашої моделі за 2 хвилини

Що таке streamlit

Streamlit — це бібліотека Python з відкритим кодом, яка дозволяє розробникам легко створювати інтерактивні веб-додатки для проєктів з обробкою даних.

Ключове про streamlit

1. **Простота**: за допомогою лише кількох рядків коду Python ви можете мати працюючий веб-додаток. Немає необхідності писати бекенд, розробляти інтерфейс або мати справу з базами даних, якщо ваша програма спеціально цього не вимагає.

2. **Інтерактивність**: Streamlit надає такі віджети, як повзунки, кнопки та введення тексту, які дозволяють користувачам взаємодіяти з даними та візуалізаціями в режимі реального часу.

3. **Інтеграція даних**: він легко інтегрується з науково-популярними бібліотеками обробки даних, такими як Pandas, NumPy, Matplotlib, Plotly та іншими.

4. **Гаряче перезавантаження**: програми Streamlit автоматично оновлюються, коли ви змінюєте свій код, тож ви можете бачити зміни в реальному часі.

5. **Налаштування**: Хоча Streamlit розроблений як простий, його також можна налаштувати під себе. Ви можете налаштувати його зовнішній вигляд за допомогою Markdown, додати інші бібліотеки Python або навіть інтегрувати розширені функції JavaScript, якщо це необхідно.

6. **Розгортання (деплоймент)**: програмами Streamlit можна легко поділитися з іншими. За допомогою таких платформ, як Streamlit sharing, або хмарних платформ, таких як AWS, GCP або Heroku, ви можете розгортати свої програми та ділитися ними з більшою аудиторією.

7. **Для чого використовується**: Streamlit зазвичай використовується для візуалізації даних, демонстрації роботи моделі машинного навчання, інструментів дослідження даних і швидкого створення прототипів програм з обробки даних.

Початок роботи

Щоб розпочати роботу зі Streamlit, необхідно встановити його через pip (`pip install streamlit`)

```
In [1]: !pip install streamlit
```

```
Collecting streamlit
  Downloading streamlit-1.23.1-py2.py3-none-any.whl (8.9 MB)
Requirement already satisfied: python-dateutil<3,>=2 in c:\users\charu\anaconda3\lib\site-packages (from streamlit) (2.8.1)
Collecting pyarrow>=4.0
  Downloading pyarrow-12.0.1-cp37-cp37m-win_amd64.whl (21.5 MB)
Collecting gitpython!=3.1.19,<4,>=3
  Downloading GitPython-3.1.40-py3-none-any.whl (190 kB)
Collecting protobuf<5,>=3.20
  Downloading protobuf-4.24.4-cp37-cp37m-win_amd64.whl (430 kB)
Requirement already satisfied: typing-extensions<5,>=4.0.1 in c:\users\charu\anaconda3\lib\site-packages (from streamlit) (4.
3.0)
Collecting toml<2
  Downloading toml-0.10.2-py2.py3-none-any.whl (16 kB)
Collecting tenacity<9,>=8.0.0
  Downloading tenacity-8.2.3-py3-none-any.whl (24 kB)
Requirement already satisfied: numpy<2,>=1 in c:\users\charu\anaconda3\lib\site-packages (from streamlit) (1.18.1)
Collecting tzlocal<5,>=1.1
  Downloading tzlocal-4.3.1-py3-none-any.whl (20 kB)
Requirement already satisfied: packaging<24,>=14.1 in c:\users\charu\anaconda3\lib\site-packages (from streamlit) (20.1)
```

Запуск програми на streamlit

Після встановлення запустити локально програму Streamlit можемо за допомогою команди:

`streamlit run your_script_name.py`.

Проєкт на GitHub

Проєкт з демо створення web app зі streamlit знайдете тут на github:

https://github.com/hpylieva/streamlit-iris-demo/tree/main