Fondamenti MLops: Parte 1

Presentazioni

clearbox AI
humans && machines

Siamo una startup ospitata dall'incubatore universitario del Politecnico di Torino.

Fondata nel 2019, stiamo sviluppando una piattaforma per la messa in produzione di modelli di machine learning, l'**AI Control Room**,

Presentazioni

Luca

Metodi e modelli matematici per auditing modelli

Andrea

Frontend, UX/UI, impacchettamento modelli

Carmine

Architettura software, infrastruttura cloud

Impostazione corso

4 sessioni da 4 ore l'una con particolare focus su esempi pratici ed esercizi dal vivo.

Canale slack per comunicazioni, link utili e domande da parte vostra.

Sempre disponibili ad aggiustare il tiro in base ai feedback → Troppo lento/troppo veloce?

Materiali e contenuti

- Libro ‘ML engineering’ di Andriy Burkov
- Blog di Martin Fowler (www.martifowler.com)
- Libro ‘Interpretable Machine Learning’ di Christophe Molnar

Contenuti selezionati e arricchiti anche in base a esperienze accumulate nel corso degli ultimi 2 anni

→

Bias verso tematiche e soluzioni di frontiera

MLOps: definizione

Applicazione concetti **DevOps** al mondo del **machine learning**.

Cosa vuol dire DevOps?

In ambito ingegneria del software con DevOps si intende un insieme di *best practices* per sviluppatori e esperti IT aventi lo scopo di efficientare lo sviluppo e la messa in produzione di strumenti informatici mantenendo alta qualità del codice scritto.

https://papers.nips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf

Hidden Technical Debt in Machine Learning Systems

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips
{dsculley,gholt,dgg,edavydov,toddphillips}@google.com
Google, Inc.

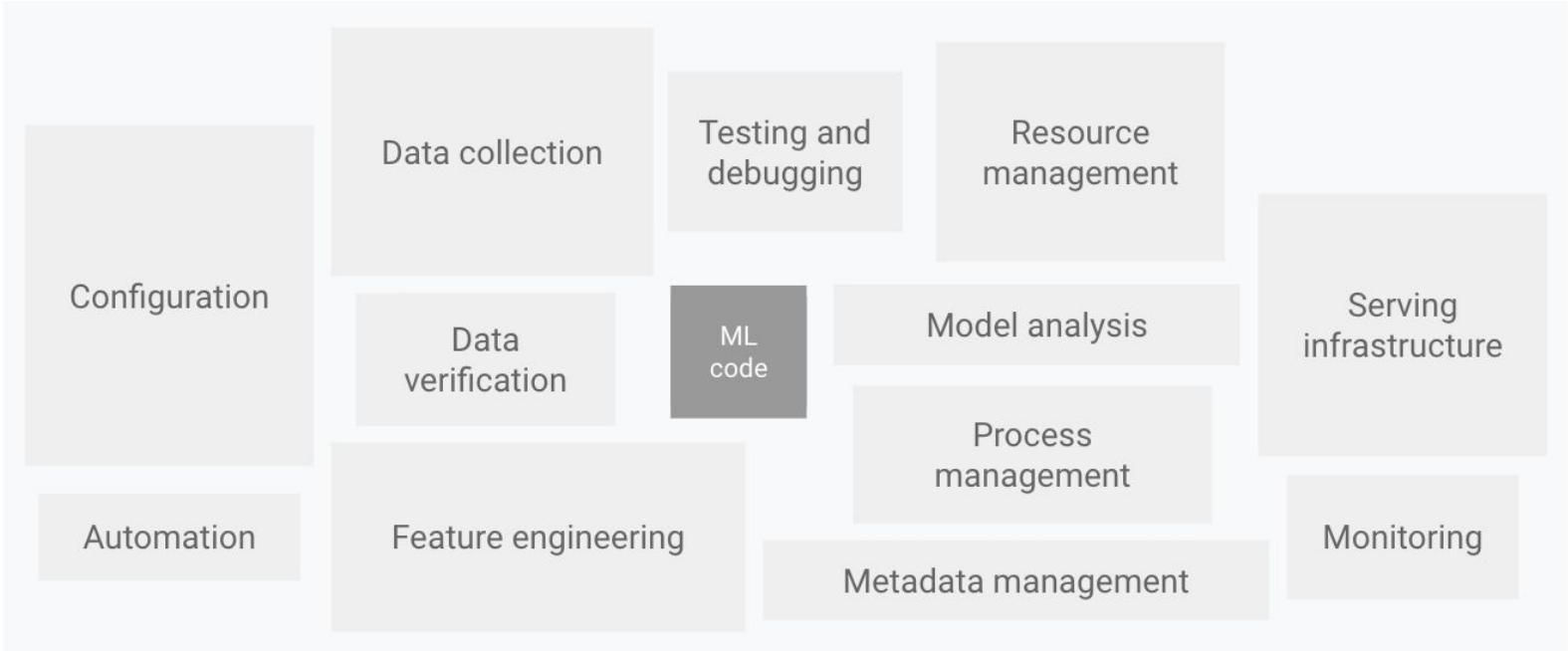
Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, Dan Dennison
{ebner,vchaudhary,mwyoung,jfcrespo,dennison}@google.com
Google, Inc.

Abstract

Machine learning offers a fantastically powerful toolkit for building useful complex prediction systems quickly. This paper argues it is dangerous to think of these quick wins as coming for free. Using the software engineering framework of *technical debt*, we find it is common to incur massive ongoing maintenance costs in real-world ML systems. We explore several ML-specific risk factors to

https://papers.nips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf

https://papers.nips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf

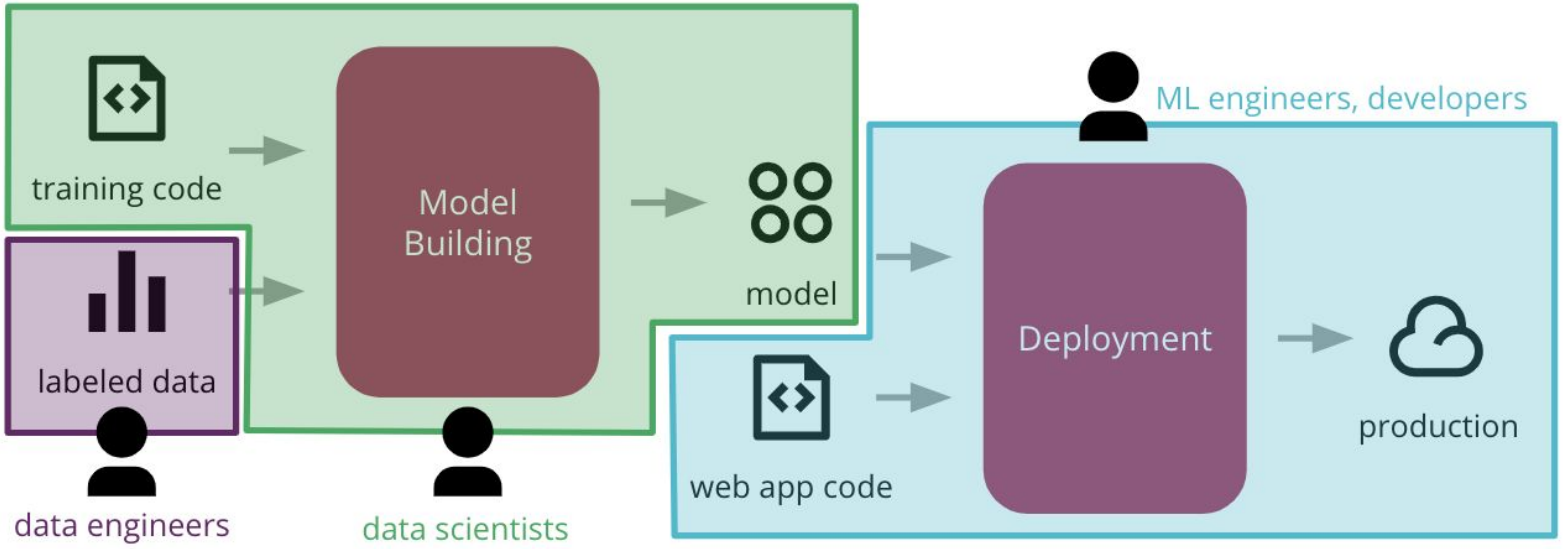


DevOps vs MLOps

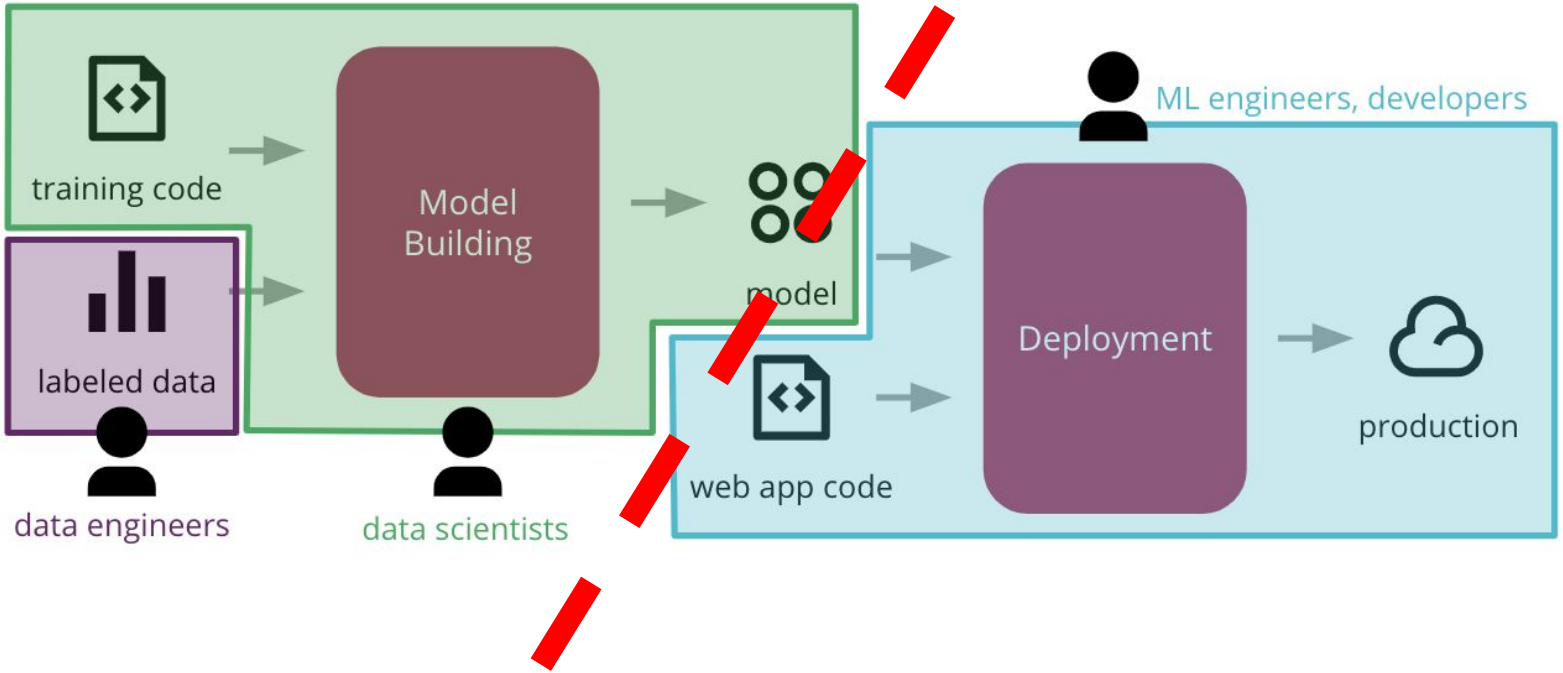
Cosa implica il considerare modelli di machine learning come strumenti software?

https://martinfowler.com/articles/cd4ml.html

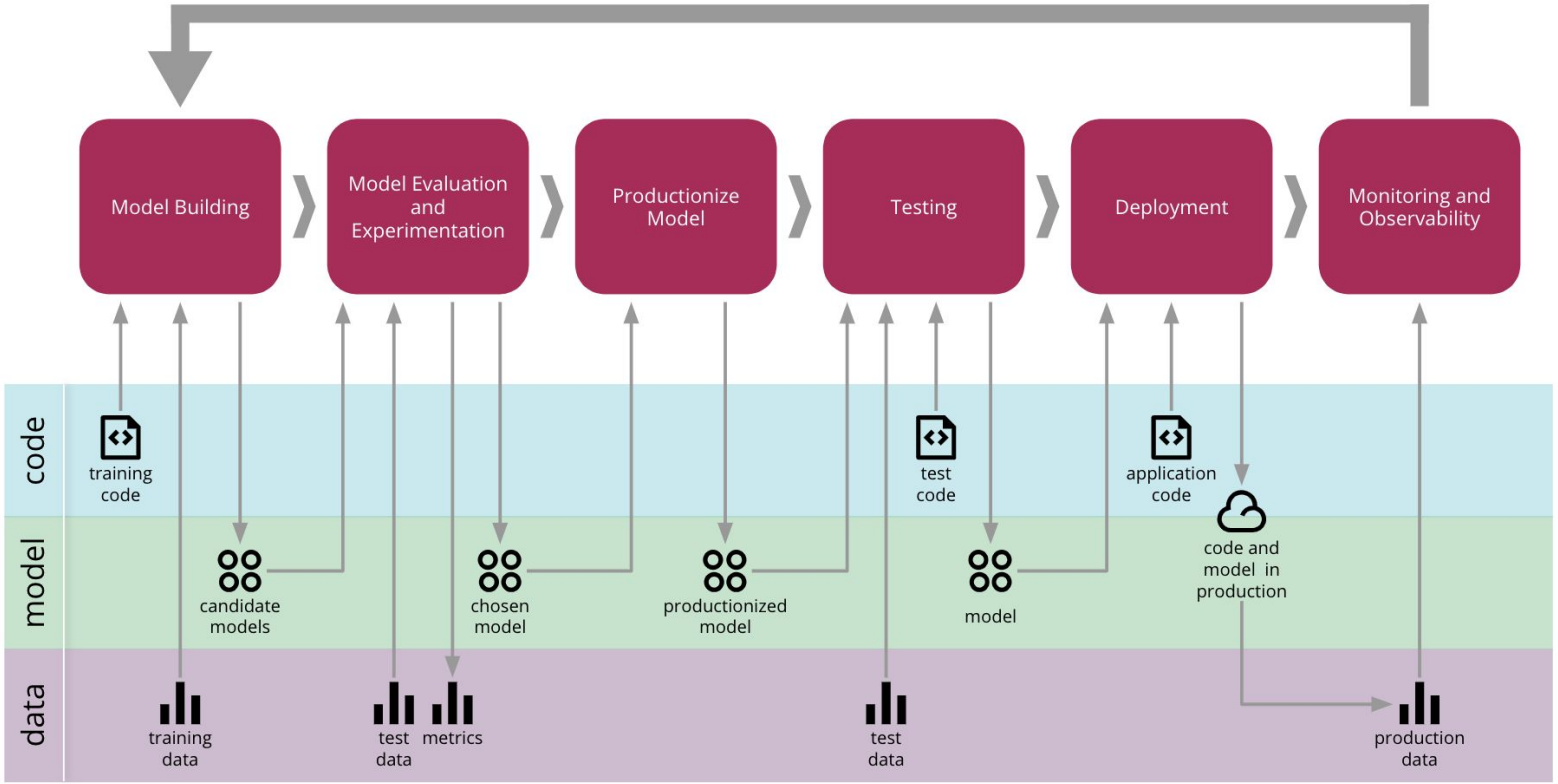
Ruoli


training code
labeled data
Model Building
model
data engineers
data scientists
ML engineers, developers
Deployment
production
web app code

Ruoli


training code
labeled data
Model Building
model
data engineers
data scientists
ML engineers, developers
Deployment
production
web app code

Machine learning in produzione



Suddivisione argomenti

Parte 1: concetti fondamentali in ambito MLops, CI/CD, riproducibilità, impacchettamento modelli.

Parte 2: Messa in produzione in cloud con esempi pratici su AWS.

Parte 3: Debugging modelli, robustezza, analisi interpretabilità e di incertezza.

Parte 4: Anomaly detection, monitoraggio modelli nel tempo e miglioramento continuo.

Il cuore del DevOps: CI/CD

Continuous Integration/ Continuous Delivery

Facilitare l’interazione tra sviluppo e produzione dando la possibilità di migliorare continuamente codice tramite piccoli interventi che vengano automaticamente messi in produzione.

Il cuore del DevOps: CI/CD

Continuous Integration/ Continuous Delivery

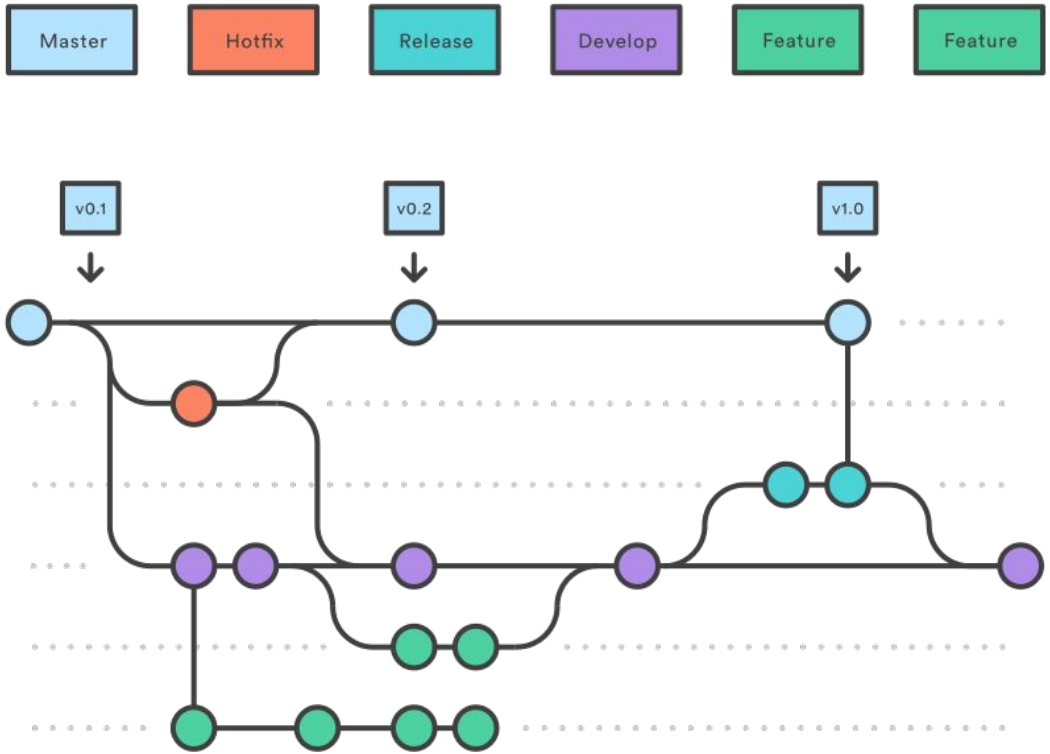
Facilitare l'interazione tra sviluppo e produzione dando la possibilità di migliorare continuamente codice tramite piccoli interventi che vengano automaticamente messi in produzione.

Piccoli interventi → Affidabili, riproducibili e utilizzabili in produzione in qualunque momento

Primo passo per il CI/CD: version control

Discussione

Primo passo per il CI/CD: version control



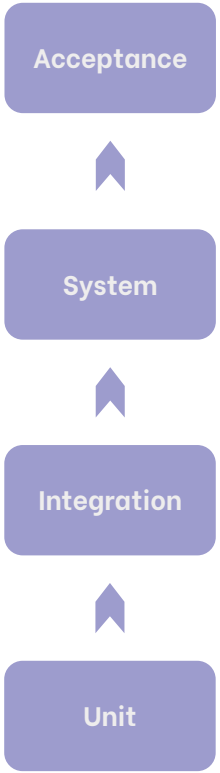
https://evolvingweb.ca/blog/ultimate-guide-agile-git-branching-workflows-drupal

Secondo passo per il CI/CD: testare, testare, testare

Ogni piccola modifica al codice deve essere sicura

Non deve rompere nulla (e nel caso rompa qualcosa si deve essere immediatamente in grado di tornare all'ultima versione funzionante)

Tipologie di test diverse a seconda di cosa si stia testando.



Esercizio

Esercizio

Problema giocattolo da usare come blueprint → Classificazione binaria (Adult Income Dataset)

Dati già divisi in :

- Train
- Validation
- Test (hold-out)

Parte 1: GitHub branches e testing con Python

1. Effettuare un fork del repo github.com/Clearbox-AI/Corso_MLOps

2. $git clone https://github.com/YOUR-USER-NAME/Corso_MLOps.git

3. Proseguire con istruzioni nel README.md

Parte 1: GitHub branches e testing con Python

Scrivere un test che verifichi che il modello serializzato:

- Non sia un ‘majority’ classifier (sia in grado di classificare più di una etichetta)
- Ottenga determinate performance sul dataset di hold out (es Precisione>80%)

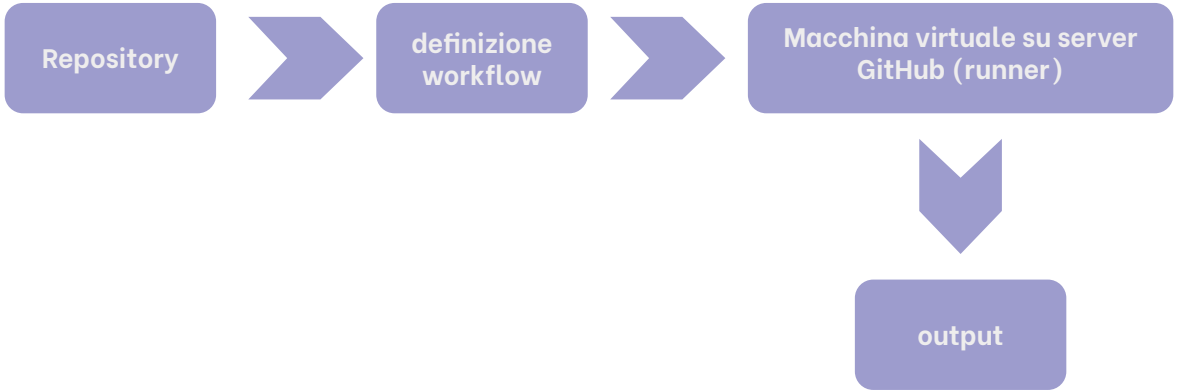
GitHub Actions

GitHub Actions

Tool per l'implementazione di pipeline automatizzate direttamente da repository GitHub

GitHub Actions

GitHub Actions



GitHub Actions

Marketplace Actions: https://github.com/marketplace?type=actions

Limiti free tier → 3000 minuti al mese di macchina virtuale per repository

```yaml
name: Test

on: [push]

jobs:
    build:
        runs-on: ubuntu-latest
        strategy:
            matrix:
                python-version: [3.7, 3.8]

        steps:
        - uses: actions/checkout@v2

        - name: Set up Python ${{ matrix.python-version }}
          uses: actions/setup-python@v2
          with:
            python-version: ${{ matrix.python-version }}

        - name: Install dependencies
          run: |
            python -m pip install --upgrade pip
            pip install -r requirements.txt
            pip install .

        - name: Pytest
          run: |
            pytest -v --maxfail=3 --cache-clear
```

Esempio:

Action che effettua test del codice ad ogni [push] su GitHub.

Esercizio: Testing tramite GitHub Action

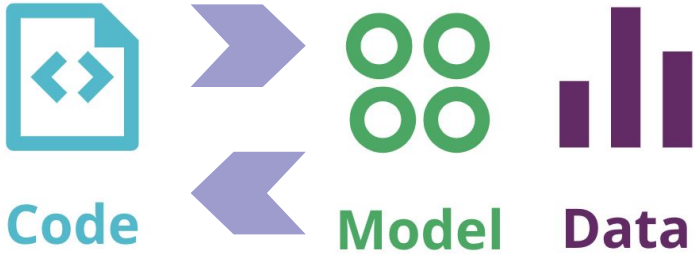
Seguire istruzioni nel README.md, ‘Creazione di una GitHub Action’

Alternative

GitLab CI/CD

Automation server classico

Il mio codice e' adeguatamente versionato... Dati e modelli?



Version control di dati e modelli: opzioni

Level 0: Nessun version control su dati e modelli (non raccomandato!)

Level 1: Dati e modelli salvati come snapshot ad ogni allenamento

Level 2: Dati e codice sono versionate come singolo asset

Level 3: Version control dedicato a dati e modelli (avanzato)

Level 4: Feature store dedicato (mega avanzato)

Version control di dati e modelli: opzioni

Level 0: Nessun version control su dati e modelli (non raccomandato!)

Level 1: Dati e modelli salvati come snapshot ad ogni allenamento

Level 2: Dati e codice sono versionate come singolo asset

Level 3: Version control dedicato a dati e modelli (avanzato)

Level 4: Feature store dedicato (mega avanzato)

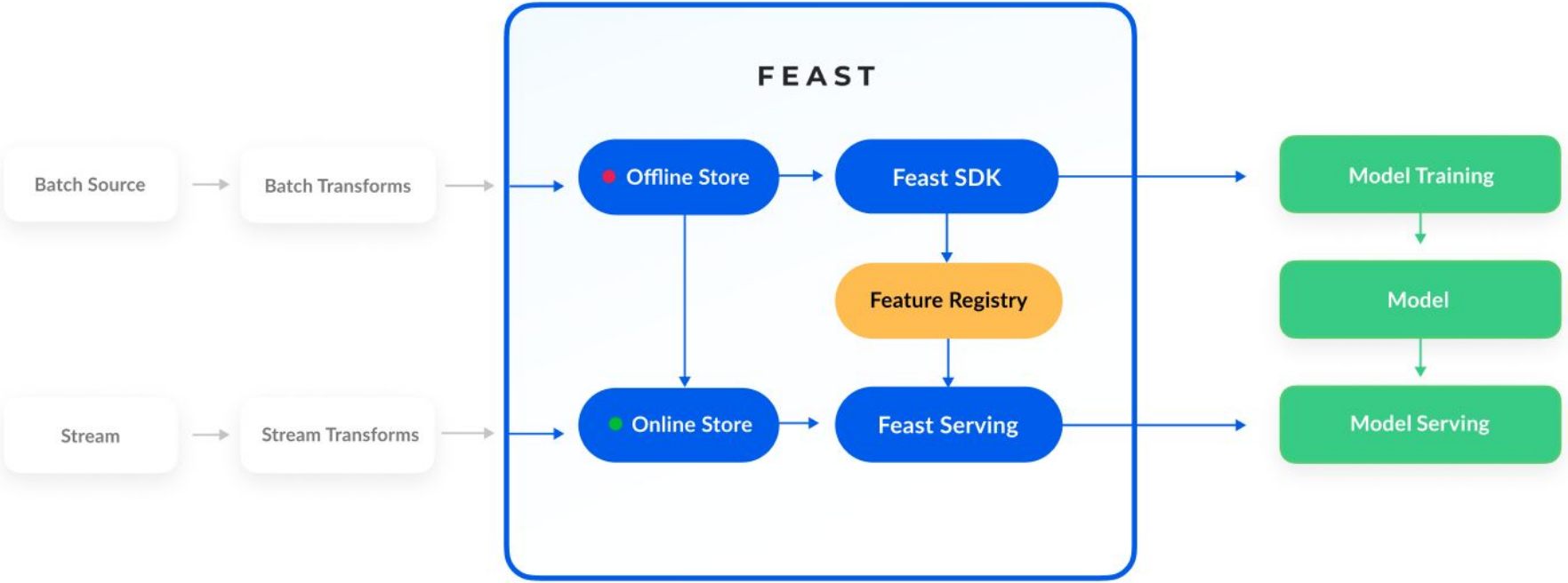
Data Version Control (esempio dal vivo)

DVC permette di versionare dati o modelli (serializzati) in maniera completamente integrata a Git.

Files possono essere salvati localmente, su storage dedicato o su cloud.

Alternativa: **Git Large File Storage** (richiede server dedicato)

Feature store

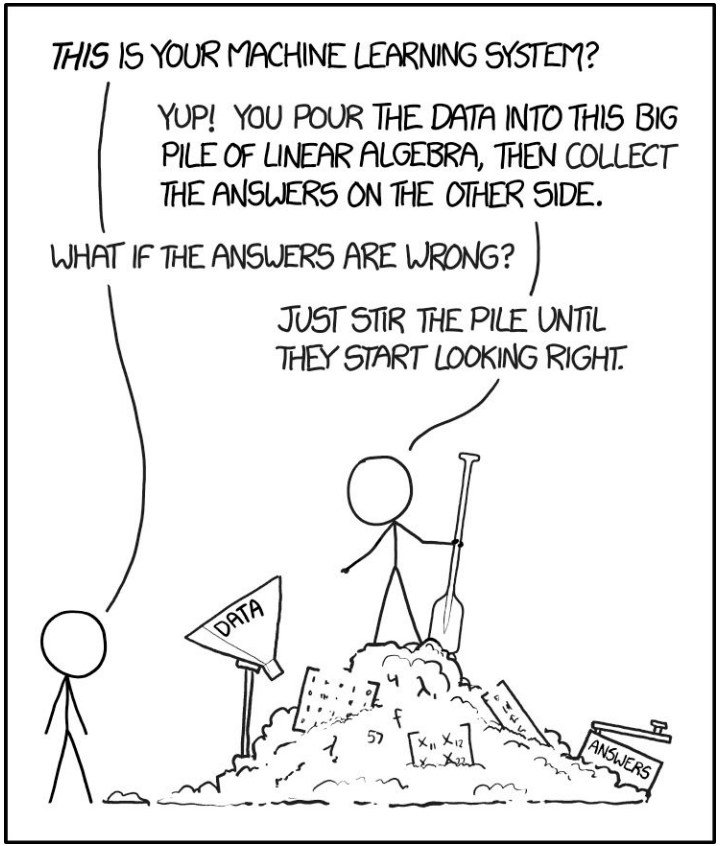


Versioning di modelli

Versioning di modelli

Lo sviluppo di modelli di machine learning sta diventando sempre più associato a ‘trial and error’

→ Problemi di riproducibilità e potenziali perdite di tempo.



mlflow

Piattaforma open source per la gestione del ciclo vita del machine learning. Permette di:

- Catalogare esperimenti e di riprodurli
- Organizzare model registries centralizzati
- Impacchettamento e messa in produzione modelli

Sviluppato e mantenuto da databricks, un bel po' di ore uomo dedicate al progetto

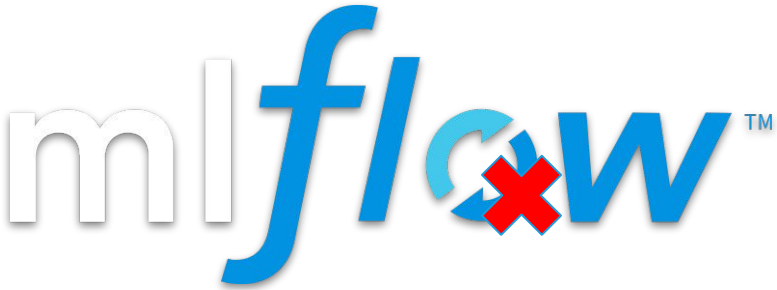
Esempio ed esercizio mlflow

Seguire istruzioni nel README.md, ‘Grid-search di iperparametri con mlflow’

Limiti di ml*flow*

Nonostante venga definito un tool end-to-end mancano ancora diverse features per quanto riguarda la parte di messa in produzione. E' invece già molto avanzato per quanto riguarda la gestione esperimenti.

Le loro librerie per impacchettare modelli producono output decisamente pesanti rispetto a soluzioni ottenute 'a mano'



Demo mlflow packaging

mlflow offre diversi tool per la messa in produzione di modelli che vanno dalla creazione di un'API, all'impacchettamento in un container al deploy su infrastruttura cloud.

Soluzione presenta notevoli vantaggi in termini di facilità d'uso caratterizzati da limiti in termini di efficienza computazionale.

Impacchettare modelli per la messa in produzione

Modelli vanno eventualmente inseriti in un contesto di produzione.

Problema: Python o R non sono esattamente efficienti da un punto di vista dell'utilizzo di risorse computazionali.

Es: Installare PyTorch in un container di produzione richiede >1Gb di spazio

Risposta: librerie dedicate all'inferenza

Open Neural Network Exchange

Libreria per **l'interoperabilità** di modelli machine learning (non solo reti neurali)

Sviluppata da Microsoft e Facebook, permette di convertire modelli esistenti in un formato leggero e performante (scritta interamente in C++). Permette ulteriori ottimizzazioni come la quantizzazione.

Esempio dal vivo: ONNX

Conversione del modello serializzato in formato ONNX tramite pacchetto sklearn2onnx.

Da modello ad applicazione

Passo finale: trasformare un modello machine learning in uno strumento software.

L'impacchettamento di un modello ML può avvenire in diversi modi a seconda del contesto in cui va a operare (web app, decision support system, processo di automazione)

Due modi di servire modelli: **batch** oppure **online**

Batch vs online serving

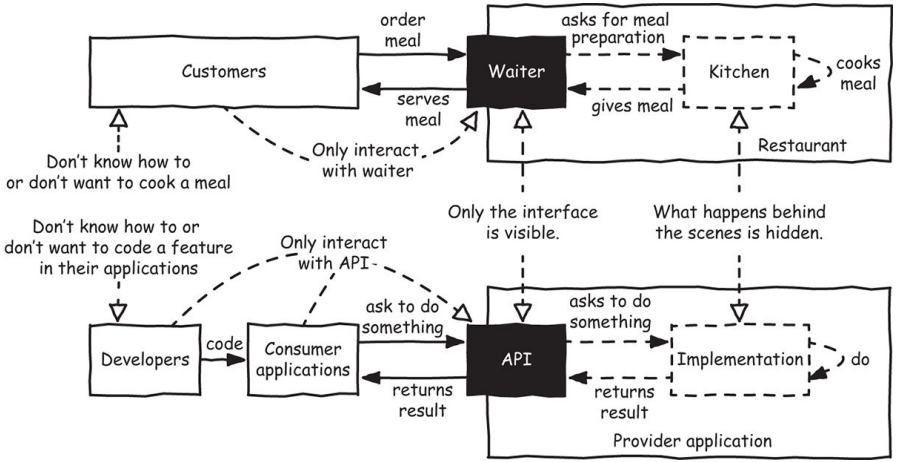
Batch: predizioni effettuate in blocchi. Solitamente rappresenta soluzione più semplice da un punto di vista di implementazione → Molti meno vincoli su tempi di esecuzione e gestione di carichi. Es: CRON task che ogni giorno processi un dataset preso da un determinato storage.

Online: modello deve fornire risposte dal vivo provenienti da uno o più utenti. Richieste diverse componenti per la gestione dello scale-up dei carichi come orchestrazione, message brokers, etc.

Online serving: FastAPI

Code

Application Programming Interface: interfaccia esposta da un software. È un'astrazione dell'implementazione sottostante (ciò che effettivamente viene eseguito all'interno del software quando l'API viene usata).



Online serving: FastAPI

Code

FastAPI

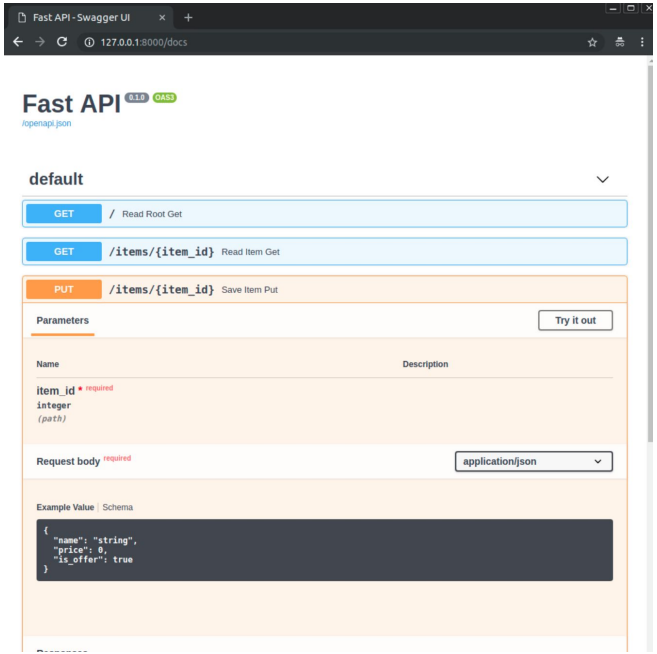
FastAPI: libreria per la creazione di API con Python 3.6+ basato sui *type hints* di Python.

- Prestazioni elevate (veloce)
- Semplice da imparare e implementare
- Documentazione automatica interattiva
- Production-ready

Online serving: FastAPI

Code

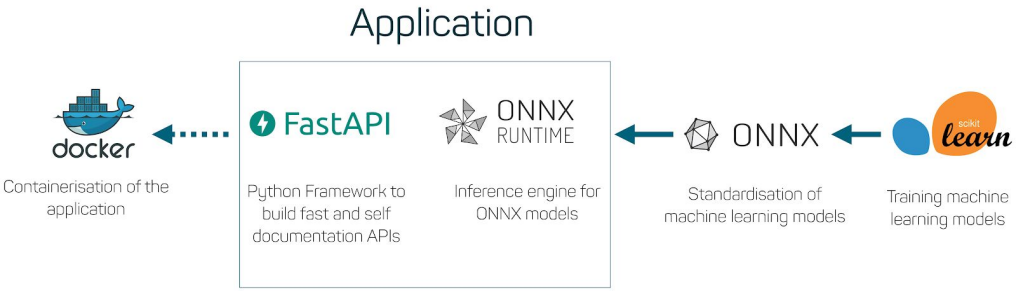
```
from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI()


class Item(BaseModel):
    name: str
    price: float
    is_offer: bool = None


@app.get("/")
def read_root():
    return {"Hello": "World"}


@app.get("/items/{item_id}")
def read_item(item_id: int, q: str = None):
    return {"item_id": item_id, "q": q}


@app.put("/items/{item_id}")
def create_item(item_id: int, item: Item):
    return {"item_name": item.name, "item_id": item_id}
```

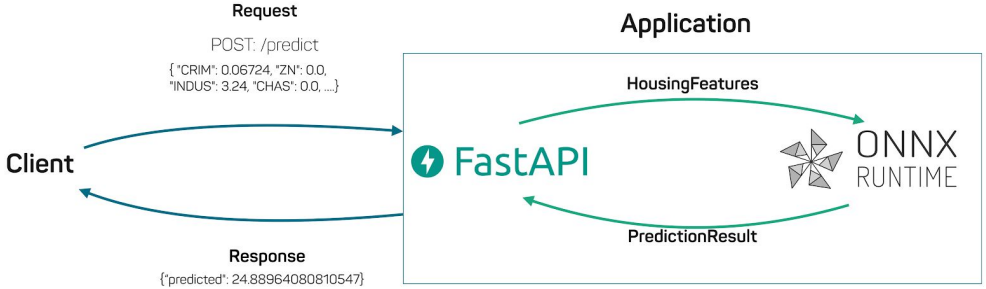


https://tiangolo.medium.com/introducing-fastapi-fdc1206d453f

Online serving: FastAPI

Code





https://medium.com/@nico.axtmann95/deploying-a-scikit-learn-model-with-onnx-und-fastapi-1af398268915

Esercizio FastAPI

Seguire istruzioni nel README.md, ‘Serving di un modello con FastAPI’

clearbox-wrapper

Iniziativa interna per l'impacchettamento di modelli.

Libreria open source basata su mlflow ma alleggerita delle componenti più pesanti e meno necessarie.

https://github.com/Clearbox-AI/clearbox-wrapper

Alcune delle esercitazioni della settimana prossima basate su questo wrapper.

Settimana prossima: deploy on cloud

Esploreremo diversi approcci per il deploy di un modello in cloud su AWS, come ad esempio:

- macchine virtuali (**EC2**)
- servizi basati su containers (**ECS**)
- soluzioni serverless. (**Lambda**)

Assignment per settimana prossima

credenziali AWS

Creare account AWS (team o individuale), potrebbe essere necessario utilizzare compute credits.

Ottenere chiavi di accesso IAM per ciascun user (access and secret key)

Installare la AWS CLI (**v2**)

esercizi codice

Impacchettare un modello a piacimento in formato **ONNX** e **clearbox-wrapper**.

(opzionale) Applicare tools discussi oggi al modello scelto.

clearbox AI
humans && machines

Thanks for Reading

Feel free to contact us:

www.clearbox.ai

luca@clearbox.ai
giovannetti@clearbox.ai

@ClearboxAI