**Section 5**

Sezione che spiega a modo come dovrebbe essere formattato un progetto in produzione, cerca di seguire le naming convention.

Usa pytest per I test (i file devono cominciare con test\_, e lanciando pytest te li lancia tutti. In un file setta le fixtures da passare in input ai vari test.

Puoi usare libreria pydantic per validare I tipi delle variabili.

Cose invece come flake8, isort, mypy ecc.. fanno automaticamente il check per te di cagate tipo linting e altro (spazi, formattazione ecc..)

Usa tox (installalo e configura tox.ini) per lanciare velocemente I tuoi vari test e testare diversi settings. Comando:

tox run -e test\_package (nome del pacchetto da lanciare)

Per buildare un package python, crea il setup.py, il MANIFEST.in e il file .toml, poi lancia:

python -m build

**Section 6**

tox run -e run

E’ settato per fare un python main.py che lancia un fast api server. Di solito faresti:

uvicorn main:app –reload

Da lì se fai /docs hai autocamaticamente accesso allo swagger che ti documenta le varie api.

Fat api è veloce come django, asincrono, può produrre documentazione delle api in automatico e gestire I tipi di dati che gli passi (gli passi I vari schemi che validano I dati, magari con anche esempi, e hai il tuo swagger già popolato.

Quando fai /predict, usa il modello che hai buildato prima (mettilo nei requirements.txt, poi importerai tutte le varie funzioni che hai creato prima)

**Section 7**

Devi mettere un folder .circleci nella root del progetto con dentro un file di config.yml. Lì dentro puoi specificare dei job che saranno quelli che il tuo processo di CI eseguirà quando lanci una build. Ad esempio installi tox e tutto e lanci tutti I tuoi vari test sul modello con tox run. Nel file di config nei workflow specifichi poi che cosa lanciare e quando triggerarli (solo quando c’è una push sul branch di master o demo, solo con tag github ecc..). Ovviamente in circleci devi specificare le varie variabili di ambiente per connetterti ai vari ambienti (kaggle, railway ecc..).

Stessa cosa per deploy del modello, installi la cli e tutto e poi fai railway up –detach

Questo comando deve essere lanciato nel folder dove hai il Procfile (che fa partire il web server con uvicorn, stile flask o django) e I requirements.txt (che installano tutte le dipendenze stile docker container, devi chiamarlo esattamente così).

Da notare che nei requirements c’è il tuo tid-regression-model (funziona perchè il tipo ha rilasciato il pacchetto pubblicamente e o vedi nel pypi). Normalmente invece dovresti scaricarlo da un private index server (ad es. Gemfury, AWS bucket, a generic web server ecc..). Per farlo nei requirements.txt aggiungi una riga con --extra-index-url=${PIP\_EXTRA\_INDEX\_URL}. In questo modo se il pacchetto non viene trovato in pypi verrà scaricato da gemfury. Questa variabile di ambiente andrà messa dentro circleci.

Quindi tu fai modifiche sul branch di demo, pushi, vai su github e ti apri una pull request su master. A questo punto vedrai che partiranno I vari job di circleci (perchè nel file hai specificato che su quei branch partissero tutti I vari test della tua applicazione e anche il deploy su railway