Domanda: Suppose now that the prototype gave good results and training must be scaled to consider all the machines in all the kitchens associated to phase 1, but reduced to a list of arepa types given in a text file (whose location is provided as additional input to your program). Briefly describe how your code would change (if you think it will) and why.

Risposta:

Per come è stato impostato lo script non sarebbe necessario utilizzare un file di testo per esplicitare il tipo di arepa ma potrebbe essere gestito tramite il file di configurazione.

Nel caso si volesse necessariamente caricare il tipo di arepa da un file di testo allora servirebbe aggiungere un file path per la posizione del file all'interno delle configurazioni e, invece di utilizzare la lista di tipi di arepa presente nelle configurazioni, andare ad utilizzare il caricamento del file di testo e la conseguente conversione di quanto caricato in lista.

Domanda: Training is complete now and it is time to put the model in production! Machines maintenance actions are decided once a day at the beginning of the day. Sketch a high level architecture of the system. How would you present the results to the users? Which are the main components and how would they interact? What about CI/CD? Assume that there is one model for each arepa type that take as input the cooking metrics from the last 7 days.

Risposta:

Sicuramente sono necessari due strumenti: un code repository (possibilmente provvisto di componenti di CI/CD utilizzabili) e un model repository.

Nel primo caso un esempio è Gitlab.

Nel secondo caso un esempio può essere MLFlow (ma anche S3 può essere forzato come tale in quale object store). Non conosco in dettaglio che funzionalità metta a disposizione AWS SageMaker.

Considerato ciò, è possibile pensare di eseguire una pipeline (o un insieme di pipeline) di CI/CD ogni mattina.

Tale pipeline, eseguita dal code repository potrebbe eseguire prima lo script appena sviluppato per aggiornare i dataset dei modelli, considerando di volta in volta gli ultimi 7 giorni in modalità rolling.

In questa narrazione sto assumendo che i dataset in ingresso allo script sviluppato per la challenge siano reperibili su una struttura relazionale (quindi interfacciandosi direttamente con un database) piuttosto che correttamente aggiornati su base giornaliera (in questo caso è da prevedere un secondo flusso dati che si occupi di questa esportazione).

Al termine dello script i dataset devono essere utilizzati per il re-training dei modelli.

Il re-training può essere effettuato con altri script python che si interfacciano con il model repository via API. In questo modo una volta trainato il modello è possibile verificarne le performance anche in relazione alle versioni precedenti presenti sul model repository e/o ad eventuali alternative (a livello di tecnica di predizione utilizzata) e selezionare quindi la versione migliore tra quelle implementate (ipotizziamo ci siano diversi tipi di modelli, ciascuno che utilizza una diversa tecnica di predizione). Altrimenti, nel caso ogni modello abbia una e una sola versione, può essere poi messo in produzione ad esempio embeddato in una web app.

In questo caso quindi, al termine della fase di re-training dei modelli, dovrebbe partire una terza pipeline che avvia il re-deploy della web app menzionata in modo tale da sostituire il modello che essa sta utilizzando. Nel caso in cui sia sufficiente sostituire il modello all'interno di specifici volumi o storage utilizzati dall'applicazione, allora sarà sufficiente che la pipeline esegua uno script responsabile di tale spostamento/sovrascrittura.