Creazione di pipeline ML pronte per la produzione su AWS

AWS Linee guida prescrittive

AWS Linee guida prescrittive: Creazione di pipeline ML pronte per la produzione su AWS

Table of Contents

Introduzione 1
..... 1
..... 3
..... 4
Utilizzo dei processi di elaborazione 4
Utilizzo della clausola main guard 6
Test unitari 6
..... 8
Utilizzo dell'SDK Step Functions 8
Estensione dello Step Functions SDK 9
..... 11
Implementazione con AWS CloudFormation 11
Modifica dell'output da Step Functions SDK 12
..... 13
..... 15
Diversi livelli di automazione 15
Piattaforme diverse per carichi di lavoro ML 16
Diversi motori per l'orchestrazione della pipeline 17
..... 18
Risorse aggiuntive 19
Cronologia dei documenti 20
..... xxi

Creazione di pipeline ML pronte per la produzione su AWS

Josiah Davis, Verdi March, Yin Song, Baichuan Sun, Chen Wu e Wei Yih Yap, Amazon Web Services ()AWS

Gennaio 2021 (cronologia dei documenti)

I progetti di machine learning (ML) richiedono un impegno significativo in più fasi che include la modellazione, l'implementazione e la produzione per fornire valore aziendale e risolvere problemi del mondo reale. In ogni fase sono disponibili numerose alternative e opzioni di personalizzazione, che rendono sempre più difficile preparare un modello di machine learning per la produzione entro i limiti delle risorse e del budget. Negli ultimi anni, in Amazon Web Services (AWS), il nostro team di Data Science ha lavorato con diversi settori industriali su iniziative di machine learning. Abbiamo identificato i punti deboli condivisi da molti AWS clienti, che derivano sia da problemi organizzativi che da sfide tecniche, e abbiamo sviluppato un approccio ottimale per fornire soluzioni ML pronte per la produzione.

Questa guida è destinata ai data scientist e agli ingegneri ML coinvolti nelle implementazioni di pipeline ML. Descrive il nostro approccio per fornire pipeline ML pronte per la produzione. La guida illustra come passare dall'esecuzione interattiva dei modelli ML (durante lo sviluppo) alla loro implementazione come parte di una pipeline (durante la produzione) per i casi d'uso del machine learning. A tal fine, abbiamo anche sviluppato una serie di modelli di esempio (vedi il progetto del progetto ML Max), per accelerare la fornitura di soluzioni ML personalizzate alla produzione, in modo da poter iniziare rapidamente senza dover fare troppe scelte di progettazione.

Panoramica di

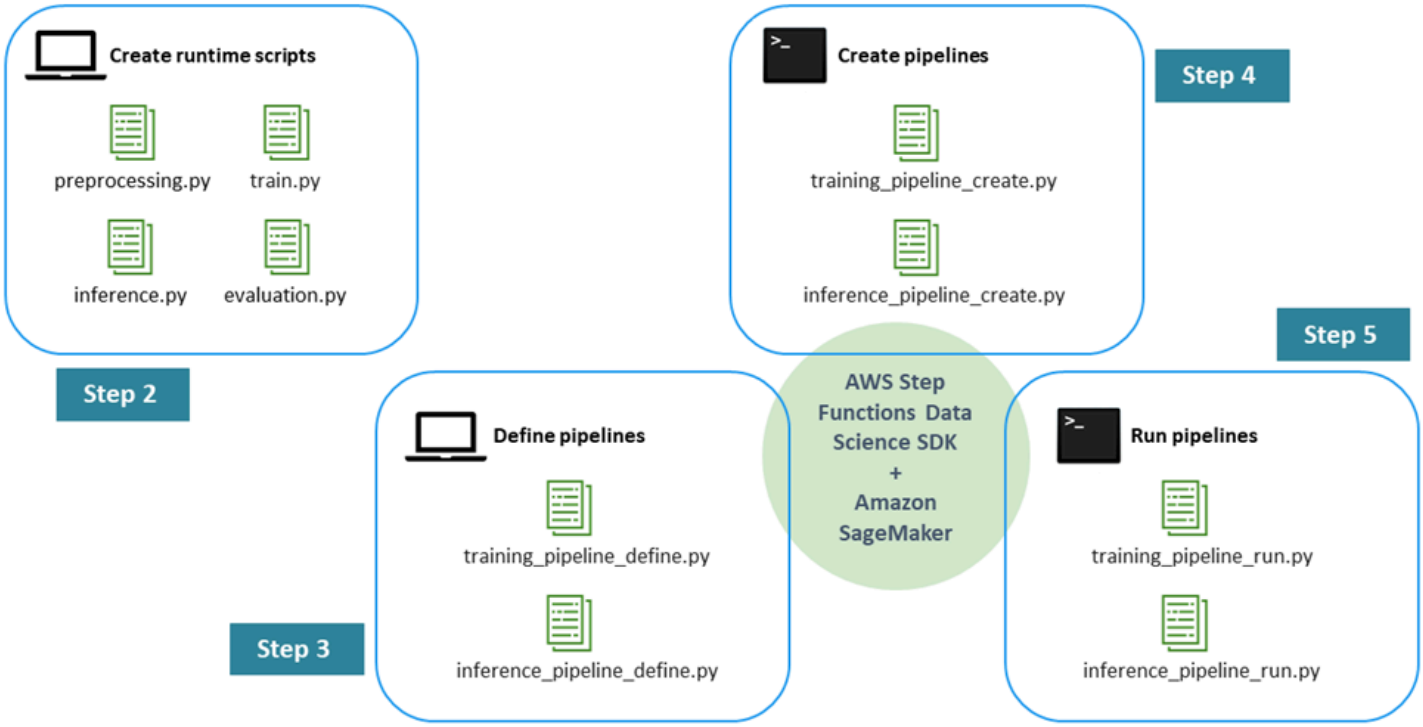
Il processo per creare una pipeline ML pronta per la produzione prevede i seguenti passaggi:

- Fase 1. Esegui EDA e sviluppa il modello iniziale: i data scientist rendono disponibili i dati grezzi in Amazon Simple Storage Service (Amazon S3), eseguono analisi esplorative dei dati (EDA), sviluppano il modello ML iniziale e ne valutano le prestazioni di inferenza. Puoi condurre queste attività in modo interattivo tramite i notebook Jupyter.
- Fase 2. Crea gli script di runtime: integri il modello con gli script Python di runtime in modo che possa essere gestito e fornito da un framework ML (nel nostro caso, Amazon AI). SageMaker Questo è il primo passo per passare dallo sviluppo interattivo di un modello autonomo alla

produzione. In particolare, si definisce separatamente la logica per la preelaborazione, la valutazione, l'addestramento e l'inferenza.

- Fase 3. Definizione della tubazione: definisci i segnaposto di input e output per ogni fase della pipeline. I relativi valori concreti verranno forniti in seguito, durante l'esecuzione (fase 5). Ti concentri sulle pipeline per la formazione, l'inferenza, la convalida incrociata e il backtest.
- Fase 4. Creazione della pipeline: crei l'infrastruttura sottostante, inclusa l'istanza della macchina a AWS Step Functions stati, in modo automatizzato (quasi con un clic), utilizzando. AWS CloudFormation
- Fase 5. Esegui la pipeline: esegui la pipeline definita nel passaggio 4. Preparate anche i metadati e i dati o le posizioni dei dati per inserire valori concreti per i input/output segnaposto definiti nel passaggio 3. Ciò include gli script di runtime definiti nel passaggio 2 e gli iperparametri del modello.
- Fase 6. Espandi la pipeline: implementerai processi di integrazione continua e distribuzione continua (CI/CD), riqualificazione automatizzata, inferenza programmata ed estensioni simili della pipeline.

Il diagramma seguente illustra le fasi principali di questo processo.



Steps for creating the training and inference pipeline

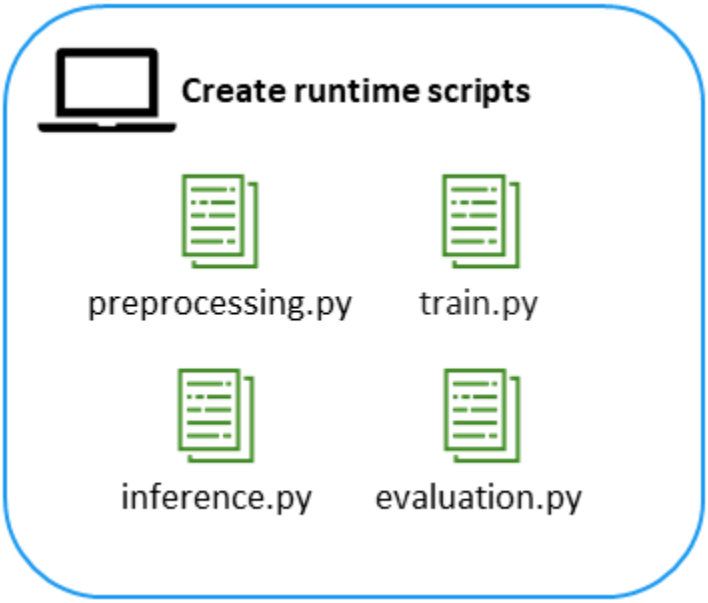
Fase 1: Esegui EDA e sviluppa il modello iniziale

In questa fase, i data scientist eseguono l'analisi esplorativa dei dati (EDA) per comprendere il caso d'uso e i dati del machine learning. Quindi sviluppano i modelli ML (ad esempio modelli di classificazione e regressione) per risolvere il problema in un determinato caso d'uso. Durante lo sviluppo del modello, il data scientist fa spesso ipotesi su input e output, come i formati dei dati, il ciclo di vita dei dati e le posizioni degli output intermedi. Queste ipotesi devono essere documentate in modo che possano essere utilizzate per la verifica durante i test unitari della fase 2.

Sebbene questa fase si concentri sullo sviluppo di modelli, i data scientist devono spesso scrivere una quantità minima di codice di supporto per la preelaborazione, la formazione, la valutazione e l'inferenza. Il data scientist dovrebbe essere in grado di eseguire questo codice nell'ambiente di sviluppo. Consigliamo inoltre di fornire argomenti di runtime opzionali in modo che questo codice di supporto possa essere configurato dinamicamente per l'esecuzione in altri ambienti senza ampie modifiche manuali. Ciò accelererà l'integrazione tra il modello e la pipeline nei passaggi 2 e 3. Ad esempio, il codice per la lettura dei dati grezzi deve essere incapsulato in funzioni in modo che i dati possano essere preelaborati in modo coerente.

Ti consigliamo di iniziare con un framework come scikit-learn,, XGBoostPyTorch, Keras o di sviluppare il modello ML e il relativo TensorFlowcodice di supporto. Ad esempio, scikit-learn è una libreria ML gratuita scritta in Python. Fornisce una convenzione API uniforme per gli oggetti e include quattro oggetti principali, estimatore, predittore, trasformatore e modello, che coprono trasformazioni leggere dei dati, supportano la progettazione di etichette e funzionalità e incapsulano le fasi di preelaborazione e modellazione. Questi oggetti aiutano a evitare la proliferazione del codice standard e impediscono la fuoriuscita di dati di convalida e test nel set di dati di addestramento. Allo stesso modo, ogni framework ML ha la propria implementazione di artefatti ML chiave e ti consigliamo di rispettare le convenzioni API del framework selezionato quando sviluppi modelli ML.

Passaggio 2. Creare gli script di runtime



In questo passaggio, integri il modello sviluppato nella fase 1 e il codice di supporto associato in una piattaforma ML per l'addestramento e l'inferenza pronti per la produzione. In particolare, ciò comporta lo sviluppo di script di runtime in modo che il modello possa essere incorporato nell'intelligenza artificiale. SageMaker Questi script Python autonomi includono funzioni di callback AI SageMaker predefinite e variabili di ambiente. Vengono eseguiti all'interno di un contenitore SageMaker AI ospitato su un'istanza Amazon Elastic Compute Cloud (Amazon EC2). La documentazione dell'SDK Amazon SageMaker AI Python fornisce informazioni dettagliate su come questi callback e la configurazione ausiliaria interagiscono per la formazione e l'inferenza. Le seguenti sezioni forniscono ulteriori consigli per lo sviluppo di script di runtime ML, in base alla nostra esperienza di lavoro con i clienti. AWS

Utilizzo dei processi di elaborazione

SageMaker L'intelligenza artificiale offre due opzioni per eseguire l'inferenza del modello in modalità batch. È possibile utilizzare un processo di elaborazione SageMaker AI o un processo di trasformazione in batch. Ogni opzione presenta vantaggi e svantaggi.

Un processo di elaborazione è costituito da un file Python che viene eseguito all'interno di un contenitore SageMaker AI. Il processo di elaborazione consiste nella logica inserita nel file Python. Presenta questi vantaggi:

- Una volta compresa la logica di base di un processo di formazione, i processi di elaborazione sono semplici da configurare e facili da capire. Condividono le stesse astrazioni dei lavori di formazione (ad esempio, l'ottimizzazione del numero di istanze e della distribuzione dei dati).
- I data scientist e gli ingegneri ML hanno il pieno controllo sulle opzioni di manipolazione dei dati.
- Il data scientist non deve gestire alcun I/O componente logico ad eccezione delle read/write funzionalità familiari.
- È leggermente più semplice eseguire i file in ambienti non basati sull'SageMaker intelligenza artificiale, il che favorisce lo sviluppo rapido e i test locali.
- Se si verifica un errore, un processo di elaborazione fallisce non appena lo script fallisce e non ci sono attese impreviste per un nuovo tentativo.

D'altra parte, i processi di trasformazione in batch sono un'estensione del concetto di endpoint SageMaker AI. In fase di esecuzione, questi job importano funzioni di callback, che poi gestiscono I/O la lettura dei dati, il caricamento del modello e l'elaborazione delle previsioni. I lavori di trasformazione in Batch presentano i seguenti vantaggi:

- Utilizzano un'astrazione di distribuzione dei dati che differisce dall'astrazione utilizzata dai lavori di formazione.
- Utilizzano lo stesso file di base e la stessa struttura di funzioni sia per l'inferenza in batch che per l'inferenza in tempo reale, il che è conveniente.
- Sono dotati di un meccanismo di tolleranza agli errori integrato basato su tentativi. Ad esempio, se si verifica un errore in un batch di record, verrà riprovato più volte prima che il processo venga terminato per errore.

Grazie alla sua trasparenza, alla sua facilità d'uso in più ambienti e all'astrazione condivisa con i processi di formazione, abbiamo deciso di utilizzare il processo di elaborazione anziché il processo di trasformazione in batch nell'architettura di riferimento presentata in questa guida.

È necessario eseguire gli script di runtime Python localmente prima di distribuirli nel cloud. In particolare, ti consigliamo di utilizzare la clausola main guard quando strutturi i tuoi script Python ed esegui il test delle unità.

Utilizzo della clausola main guard

Usa una clausola main guard per supportare l'importazione dei moduli e per eseguire lo script Python. L'esecuzione di script Python singolarmente è utile per il debug e l'isolamento dei problemi nella pipeline ML. È consigliabile eseguire le operazioni seguenti:

- Usa un parser di argomenti nei file di elaborazione Python per input/output specificare i file e le loro posizioni.
- Fornisci una guida principale e funzioni di test per ogni file Python.
- Dopo aver testato un file Python, incorporalo nelle diverse fasi della pipeline ML, indipendentemente dal fatto che tu stia utilizzando un AWS Step Functions modello o un processo di elaborazione SageMaker AI.
- Utilizza le istruzioni Assert nelle sezioni critiche dello script per facilitare il test e il debug. Ad esempio, è possibile utilizzare un'istruzione Assert per garantire che il numero di funzionalità del set di dati sia costante dopo il caricamento.

Test unitari

Il test unitario degli script di runtime scritti per la pipeline è un'attività importante che viene spesso ignorata nello sviluppo di pipeline ML. Questo perché l'apprendimento automatico e la scienza dei dati sono campi relativamente nuovi e hanno tardato ad adottare pratiche di ingegneria del software consolidate come i test unitari. Poiché la pipeline ML verrà utilizzata nell'ambiente di produzione, è essenziale testare il codice della pipeline prima di applicare il modello ML alle applicazioni del mondo reale.

Il test unitario dello script di runtime offre anche i seguenti vantaggi esclusivi per i modelli ML:

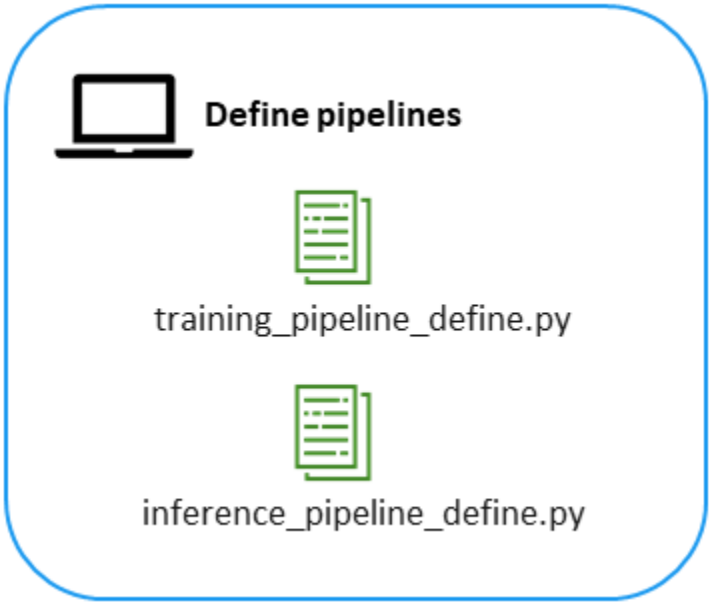
- Impedisce trasformazioni impreviste dei dati. La maggior parte delle pipeline ML comporta molte trasformazioni di dati, quindi è fondamentale che queste trasformazioni funzionino come previsto.
- Convalida la riproducibilità del codice. Qualsiasi casualità nel codice può essere rilevata mediante test unitari con diversi casi d'uso.
- Rafforza la modularità del codice. I test unitari sono generalmente associati alla misura di copertura del test, che è il grado in cui una particolare suite di test (una raccolta di casi di test) esegue il codice sorgente di un programma. Per ottenere un'elevata copertura dei test, gli sviluppatori modularizzano il codice, perché è difficile scrivere test unitari per una grande quantità di codice senza suddividerlo in funzioni o classi.

- Impedisce l'introduzione di codice o errori di bassa qualità nella produzione.

Ti consigliamo di utilizzare un framework di unit test maturo come pytest per scrivere i casi di unit test, perché è più facile gestire test unitari completi all'interno di un framework.

> **Important**
>
> I test unitari non possono garantire che tutti gli corner case vengano testati, ma possono aiutarvi a evitare in modo proattivo gli errori prima di implementare il modello. Si consiglia di monitorare il modello anche dopo l'implementazione, per garantire l'eccellenza operativa.

Fase 3. Definire la pipeline



In questo passaggio vengono definite la sequenza e la logica delle azioni che la pipeline eseguirà. Ciò include i passaggi discreti, nonché i relativi ingressi e uscite logici. Ad esempio, qual è lo stato dei dati all'inizio della pipeline? Proviene da più file con diversi livelli di granularità o da un singolo file flat? Se i dati provengono da più file, è necessario un unico passaggio per tutti i file o passaggi separati per ogni file per definire la logica di preelaborazione? La decisione dipende dalla complessità delle fonti di dati e dalla misura in cui vengono preelaborate.

Nella nostra implementazione di riferimento, utilizziamo AWS Step Functions, che è un orchestratore di funzioni senza server, per definire le fasi del flusso di lavoro. Tuttavia, il framework ML Max supporta anche altri sistemi di pipeline o macchine a stati come Apache AirFlow (vedi la sezione Diversi motori per l'orchestrazione delle pipeline) per promuovere lo sviluppo e l'implementazione di pipeline ML.

Utilizzo dell'SDK Step Functions

Per definire la pipeline ML, utilizziamo innanzitutto l'API Python di alto livello fornita da AWS Step Functions Data Science SDK (Step Functions SDK) per definire due componenti chiave della pipeline: passaggi e dati. Se si pensa a una pipeline come a un grafo aciclico diretto (DAG), i passaggi rappresentano i nodi sul grafico e i dati vengono visualizzati come bordi diretti che collegano un nodo (step) al successivo. Esempi tipici di fasi di apprendimento automatico includono

la preelaborazione, la formazione e la valutazione. Step Functions SDK fornisce una serie di passaggi integrati (come TrainingStep) che è possibile utilizzare. Esempi di dati includono input, output e molti set di dati intermedi prodotti da alcune fasi della pipeline. Quando si progetta una pipeline ML, non si conoscono i valori concreti degli elementi di dati. È possibile definire segnaposto di dati che fungono da modello (in modo simile ai parametri delle funzioni) e contengono solo il nome dell'elemento di dati e i tipi di dati primitivi. In questo modo, puoi progettare un progetto di pipeline completo senza conoscere in anticipo i valori concreti dei dati che viaggiano sul grafico. A tale scopo, puoi utilizzare le classi placeholder in Step Functions SDK per modellare in modo esplicito questi modelli di dati.

Le pipeline ML richiedono anche parametri di configurazione per eseguire un controllo dettagliato sul comportamento di ogni fase ML. Questi segnaposto di dati speciali sono chiamati segnaposto parametrici. Molti dei loro valori sono sconosciuti quando si definisce la pipeline. Esempi di segnaposto per parametri includono i parametri relativi all'infrastruttura definiti durante la progettazione della pipeline (ad esempio, Regione AWS l'URL dell'immagine del contenitore) e i parametri relativi alla modellazione ML (come gli iperparametri) che definisci quando esegui la pipeline.

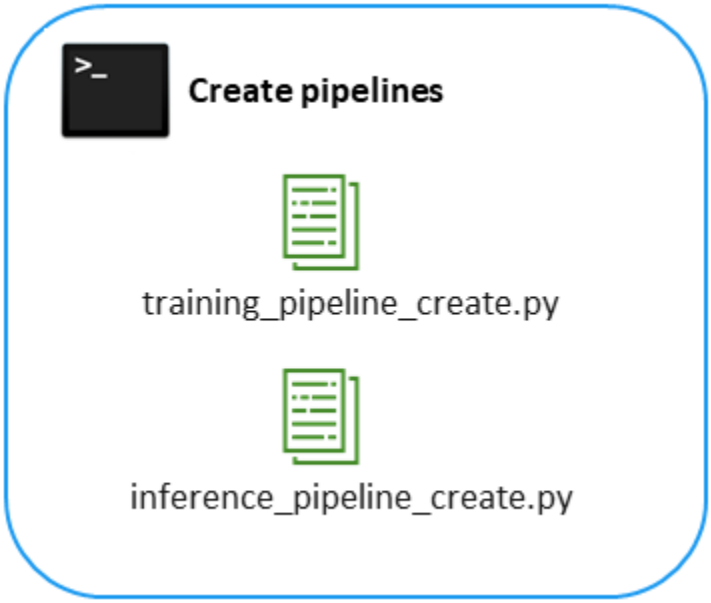
Estensione dello Step Functions SDK

Nella nostra implementazione di riferimento, uno dei requisiti era quello di separare le definizioni delle pipeline ML dalla creazione e distribuzione di pipeline ML concrete utilizzando impostazioni di parametri specifiche. Tuttavia, alcuni dei passaggi integrati in Step Functions SDK non ci hanno permesso di passare tutti questi parametri segnaposto. Invece, ci si aspettava che i valori dei parametri fossero ottenuti direttamente durante la fase di progettazione della pipeline tramite chiamate API di configurazione SageMaker AI. Ciò funziona bene se l'ambiente SageMaker AI in fase di progettazione è identico all'ambiente di runtime SageMaker AI, ma ciò accade raramente nelle impostazioni del mondo reale. Un abbinamento così stretto tra tempo di progettazione e runtime della pipeline e il presupposto che l'infrastruttura della piattaforma ML rimanga costante, ostacola in modo significativo l'applicabilità della pipeline progettata. In effetti, la pipeline ML si interrompe immediatamente quando la piattaforma di implementazione sottostante subisce anche la minima modifica.

Per superare questa sfida e produrre una solida pipeline ML (che volevamo progettare una sola volta ed eseguire ovunque), abbiamo implementato i nostri passaggi personalizzati estendendo alcuni dei passaggi integrati, tra cui, e. TrainingStepModelStepTransformerStep Queste estensioni sono fornite nel progetto ML Max. Le interfacce di questi passaggi personalizzati supportano molti più segnaposto

di parametri che possono essere compilati con valori concreti durante la creazione della pipeline o quando la pipeline è in esecuzione.

Passaggio 4. Crea la pipeline



Dopo aver definito la pipeline in modo logico, è il momento di creare l'infrastruttura di supporto della pipeline. Questo passaggio richiede almeno le seguenti funzionalità:

- Storage, per ospitare e gestire gli input e gli output della pipeline, inclusi codice, artefatti del modello e dati utilizzati nelle esecuzioni di addestramento e inferenza.
- Elaborazione (GPU o CPU), per la modellazione e l'inferenza, nonché per la preelaborazione e la post-elaborazione dei dati.
- Orchestrazione, per gestire le risorse utilizzate e pianificare eventuali esecuzioni regolari. Ad esempio, il modello potrebbe essere riqualificato periodicamente non appena diventano disponibili nuovi dati.
- Registrazione e invio di avvisi, per monitorare l'accuratezza del modello di pipeline, per l'utilizzo delle risorse e per la risoluzione dei problemi.

Implementazione con AWS CloudFormation

Per creare la pipeline che abbiamo utilizzato AWS CloudFormation, che è un AWS servizio per l'implementazione e la gestione dell'infrastruttura come codice. I AWS CloudFormation modelli includono la definizione Step Functions creata nel passaggio precedente con Step Functions SDK. Questo passaggio include la creazione dell'istanza AWS-managed Step Functions, chiamata

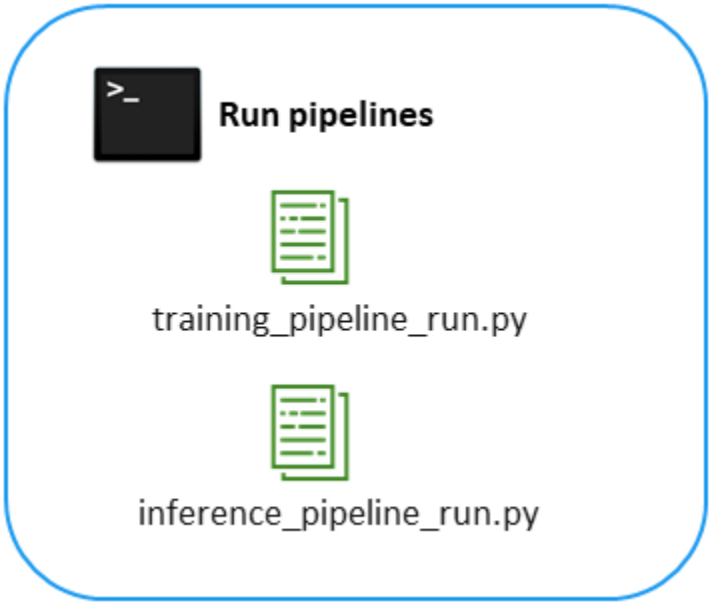
macchina a stati Step Functions. In questa fase non vengono create risorse per la formazione e l'inferenza, poiché i lavori di formazione e inferenza vengono eseguiti su richiesta, solo quando sono necessari, come lavori di SageMaker intelligenza artificiale. Questo passaggio include anche la creazione di ruoli AWS Identity and Access Management (IAM) per eseguire Step Functions, eseguire l' SageMaker intelligenza artificiale e leggere e scrivere da Amazon S3.

Modifica dell'output da Step Functions SDK

Abbiamo dovuto apportare alcune piccole modifiche all' CloudFormation output della sezione precedente. Abbiamo usato una semplice corrispondenza di stringhe in Python per fare quanto segue:

- Abbiamo aggiunto la logica per creare la `Parameters` sezione del CloudFormation modello. Questo perché vogliamo creare due ruoli e definire il nome della pipeline come parametro insieme all'ambiente di distribuzione. Questo passaggio copre anche eventuali risorse e ruoli aggiuntivi che potreste voler creare, come illustrato nel passaggio 6.
- Abbiamo riformattato tre campi in modo che abbiano il `!Sub` prefisso e le virgolette richiesti in modo che possano essere aggiornati dinamicamente come parte del processo di distribuzione:
  - La `StateMachineName` proprietà, che dà il nome alla macchina a stati.
  - La `DefinitionString` proprietà, che definisce la macchina a stati.
  - La `RoleArn` proprietà, che viene restituita dalla macchina a stati.

Fase 5. Esegui la pipeline



Questo passaggio esegue la pipeline di addestramento o inferenza creata negli CloudFormation stack nel passaggio 4. La pipeline non può essere eseguita finché i parametri segnaposto interni non sono stati compilati con valori concreti. Questa azione di assegnazione di valori ai parametri segnaposto è l'attività principale della fase 5. I parametri segnaposto di esempio includono:

- La posizione dei set di dati di input, output e intermedi
- La posizione in Amazon S3 degli script di runtime e di altro codice di preelaborazione o valutazione sviluppato nella fase 2 (ad esempio, `sm_submit_url` per la pipeline di formazione)
- Il nome della regione AWS

È necessario assicurarsi che questi valori di percorso puntino a dati o codice validi prima di eseguire la pipeline. Ad esempio, se compili il parametro placeholder che rappresenta l'URL Amazon S3 degli script di runtime Python, devi caricare quegli script su quell'URL. La persona che gestisce la pipeline è responsabile del controllo della coerenza e del caricamento dei dati. Le persone che definiscono o creano la pipeline non devono preoccuparsi di nulla di tutto ciò.

A seconda della maturità della pipeline, questo passaggio può essere automatizzato per essere eseguito su base regolare (settimanale o mensile). L'automazione richiede anche un monitoraggio affidabile, che è un'area importante ma non rientra nell'ambito di questa guida. Per lo svolgimento della pipeline di formazione, sarebbe opportuno monitorare le metriche di valutazione. Per quanto

riguarda la pipeline di inferenza, sarebbe opportuno monitorare la deriva della distribuzione dei dati in ingresso e, se possibile, raccogliere periodicamente etichette e misurare la deriva nell'accuratezza della previsione. Questi record delle sessioni di addestramento e inferenza devono essere registrati in un database per essere analizzati in un secondo momento.

Fase 6. Espandi la pipeline

Questa guida spiega come iniziare a creare pipeline ML AWS rapidamente, con un'architettura concreta. Esistono altre considerazioni per la maturazione della pipeline, come la gestione dei metadati, il tracciamento degli esperimenti e il monitoraggio. Si tratta di argomenti importanti che non rientrano nell'ambito di questa guida. Nelle sezioni seguenti viene illustrato un altro aspetto della gestione delle pipeline, ovvero l'automazione delle pipeline.

Diversi livelli di automazione

Sebbene sia possibile configurare manualmente una pipeline di formazione nella console di SageMaker intelligenza artificiale, in pratica consigliamo di ridurre al minimo i punti di contatto manuali nell'implementazione delle pipeline di formazione ML per garantire che i modelli ML vengano implementati in modo coerente e ripetuto. A seconda delle esigenze e dei problemi aziendali da risolvere, è possibile determinare e implementare una strategia di implementazione su tre livelli: semiautomatizzata, completamente automatizzata e completamente gestita.

- Semiautomatico: per impostazione predefinita, i passaggi descritti nella sezione precedente seguono un approccio semiautomatico, poiché implementano la pipeline di formazione e inferenza utilizzando modelli. CloudFormation Ciò contribuisce a garantire la riproducibilità della pipeline e consente di modificarla e aggiornarla facilmente.
- Completamente automatizzato: un'opzione più avanzata consiste nell'utilizzare l'integrazione continua e l'implementazione continua (CI/CD) to the development, staging, and production environments. Incorporating CI/CDle pratiche di implementazione della pipeline di formazione possono garantire che l'automazione includa la tracciabilità e i parametri di qualità).
- Completamente gestito: in definitiva, è possibile sviluppare un sistema completamente gestito in modo da poter implementare una pipeline di formazione ML con una serie di manifesti semplici e il sistema può configurare e coordinare automaticamente i servizi richiesti. AWS

In questa guida, abbiamo scelto di presentare un'architettura concreta. Tuttavia, ci sono tecnologie alternative che puoi prendere in considerazione. Le due sezioni successive illustrano alcune scelte alternative per la piattaforma e il motore di orchestrazione.

Piattaforme diverse per carichi di lavoro ML

Amazon SageMaker AI è il servizio AWS gestito per la formazione e la gestione di modelli di machine learning. Molti utenti apprezzano la sua vasta gamma di funzionalità integrate e le numerose opzioni che offre per l'esecuzione di carichi di lavoro ML. SageMaker L'intelligenza artificiale è particolarmente utile se hai appena iniziato a implementare il machine learning nel cloud. Le caratteristiche principali dell' SageMaker IA includono:

- Tracciabilità integrata (inclusi etichettatura, formazione, tracciamento dei modelli, ottimizzazione e inferenza).
- Opzioni integrate con un solo clic per l'addestramento e l'inferenza con un'esperienza minima in Python e ML.
- Ottimizzazione avanzata degli iperparametri.
- Support per tutti i principali framework di intelligenza artificiale e machine learning (ML/AI) e contenitori Docker personalizzati.
- Funzionalità di monitoraggio integrate.
- Monitoraggio integrato delle cronologie, inclusi lavori di formazione, processi di elaborazione, processi di trasformazione in batch, modelli, endpoint e ricercabilità. Alcune cronologie, come la formazione, l'elaborazione e la trasformazione in batch, sono immutabili e possono essere aggiunte solo.

Una delle alternative all'utilizzo dell'IA è. SageMaker AWS Batch AWS Batch offre un livello inferiore di controllo sull'elaborazione e l'orchestrazione per l'ambiente, ma non è progettato su misura per l'apprendimento automatico. Alcune delle sue caratteristiche principali includono:

- Out-of-the-box scalabilità automatica delle risorse di elaborazione in base al carico di lavoro.
- Out-of-the-box supporto per la priorità del lavoro, i nuovi tentativi e le dipendenze lavorative.
- Approccio basato sulla coda che supporta la creazione di lavori ricorrenti e su richiesta.
- Support per carichi di lavoro di CPU e GPU. La capacità di utilizzare la GPU per la creazione di modelli di machine learning è fondamentale, poiché la GPU può accelerare notevolmente il processo di formazione, specialmente per i modelli di deep learning.
- Capacità di definire un'Amazon Machine Image (AMI) personalizzata per l'ambiente di calcolo.

Diversi motori per l'orchestrazione della pipeline

Il secondo componente principale è il livello di orchestrazione della pipeline. AWS fornisce Step Functions per un'esperienza di orchestrazione completamente gestita. Un'alternativa popolare a Step Functions è Apache Airflow. Quando prendi una decisione tra le due, considera quanto segue:

- Infrastruttura richiesta: AWS Step Functions è un servizio completamente gestito e senza server, mentre Airflow richiede la gestione dell'infrastruttura propria ed è basato su software open source. Di conseguenza, Step Functions offre un'elevata disponibilità immediata, mentre l'amministrazione di Apache Airflow richiede passaggi aggiuntivi.
- Funzionalità di pianificazione: sia Step Functions che Airflow offrono funzionalità comparabili.
- Funzionalità di visualizzazione e interfaccia utente: sia Step Functions che Airflow offrono funzionalità comparabili.
- Passaggio di variabili all'interno del grafico computazionale: Step Functions fornisce funzionalità limitate per l'utilizzo AWS Lambda delle funzioni, mentre Airflow fornisce XCom interfacce.
- Utilizzo: Step Functions è molto popolare tra AWS i clienti e Airflow è stato ampiamente adottato dalla comunità di ingegneria dei dati.

Conclusioni

Man mano che l'apprendimento automatico passa da una disciplina di ricerca a un campo applicato, abbiamo registrato una crescita annua del 25% nello sviluppo, nell'implementazione e nella gestione di pipeline ML in vari settori. Il valore aziendale del machine learning si realizza attraverso le operazioni e le pipeline di day-to-day machine learning, che a loro volta guidano la ricerca e lo sviluppo di modelli e algoritmi di machine learning. Tuttavia, l'implementazione del machine learning in produzione presenta numerose sfide, poiché intreccia attività e artefatti significativamente diversi, come la gestione, l'elaborazione, l'analisi, la modellazione, la verifica e la sicurezza dei dati. Attraverso numerose collaborazioni in ambito AI/ML con AWS i clienti, il nostro team di Data Science ha osservato che una sfida fondamentale è la mancanza di un end-to-end flusso di lavoro che fornisca una serie di modelli per fondere o separare in modo ottimale diverse attività e artefatti di machine learning. DevOps In questa guida, abbiamo presentato il flusso di lavoro ML Max per risolvere questo problema urgente. ML Max fornisce step-by-step linee guida e una serie di modelli di programmazione. L'obiettivo è consentire una transizione rapida ed economica da una fase di sviluppo del modello interattivo a una configurazione di pipeline ML completa e scalabile pronta per la produzione.

Risorse aggiuntive

Guide e modelli correlati

- Quantificazione dell'incertezza nei sistemi di deep learning
- Esegui la migrazione dei carichi di lavoro ML (build, training e deploy) su Amazon SageMaker AI utilizzando Developer Tools AWS
- AWS Sito web Prescriptive Guidance

AWS servizi

- AWS Batch
- AWS CloudFormation
- IAM
- Amazon SageMaker AI
- AWS Step Functions

AWS risorse generali

- AWS documentazione
- AWS riferimento generale
- AWS glossario

Cronologia dei documenti

La tabella seguente descrive le modifiche significative apportate a questa guida. Per ricevere notifiche sugli aggiornamenti futuri, puoi abbonarti a un feed RSS.

Modifica	Descrizione	Data
Pubblicazione iniziale	—	29 gennaio 2021

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.