Erstellung produktionsreifer ML-Pipelines auf AWS

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Erstellung produktionsreifer ML-Pipelines auf AWS

Table of Contents

Erstellung produktionsreifer ML-Pipelines auf AWS

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

Januar 2021 (Geschichte der Dokumente)

Projekte für maschinelles Lernen (ML) erfordern einen erheblichen, mehrstufigen Aufwand, der Modellierung, Implementierung und Produktion umfasst, um einen Mehrwert für das Unternehmen zu erzielen und reale Probleme zu lösen. In jedem Schritt stehen zahlreiche Alternativen und Anpassungsoptionen zur Verfügung. Dadurch wird es immer schwieriger, ein ML-Modell für die Produktion innerhalb der Grenzen Ihrer Ressourcen und Ihres Budgets vorzubereiten. In den letzten Jahren hat unser Data Science-Team bei Amazon Web Services (AWS) mit verschiedenen Branchen an ML-Initiativen gearbeitet. Wir haben Probleme identifiziert, die viele AWS Kunden gemeinsam haben und die sowohl auf organisatorische Probleme als auch auf technische Herausforderungen zurückzuführen sind, und wir haben einen optimalen Ansatz für die Bereitstellung produktionsreifer ML-Lösungen entwickelt.

Dieser Leitfaden richtet sich an Datenwissenschaftler und ML-Ingenieure, die an ML-Pipeline-Implementierungen beteiligt sind. Er beschreibt unseren Ansatz zur Bereitstellung produktionsreifer ML-Pipelines. In diesem Leitfaden wird erläutert, wie Sie von der interaktiven Ausführung von ML-Modellen (während der Entwicklung) zur Bereitstellung als Teil einer Pipeline (während der Produktion) für Ihren ML-Anwendungsfall übergehen können. Zu diesem Zweck haben wir auch eine Reihe von Beispielvorlagen entwickelt (siehe das ML Max-Projektprojekt), um die Bereitstellung benutzerdefinierter ML-Lösungen für die Produktion zu beschleunigen, sodass Sie schnell loslegen können, ohne zu viele Designentscheidungen treffen zu müssen.

-Übersicht

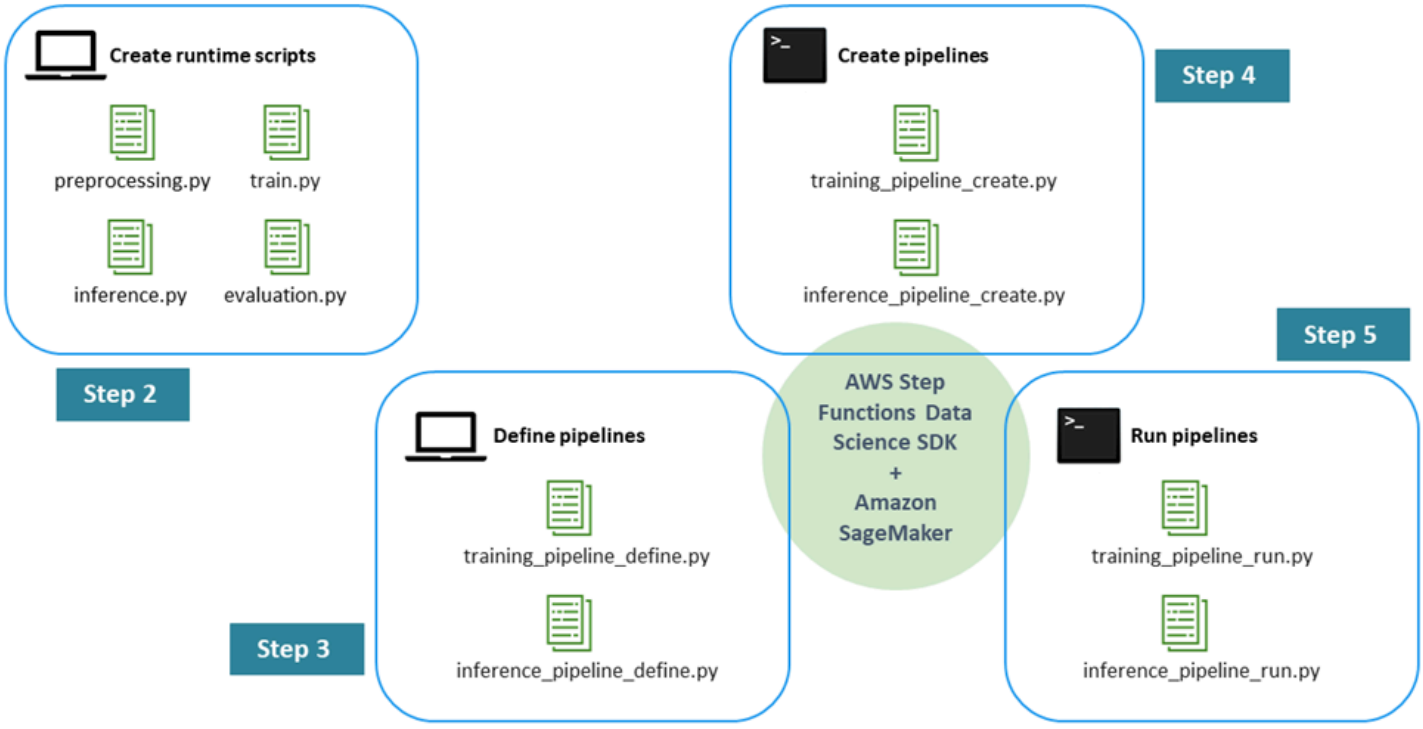
Der Prozess zur Erstellung einer produktionsbereiten ML-Pipeline besteht aus den folgenden Schritten:

- Schritt 1. EDA durchführen und das erste Modell entwickeln — Datenwissenschaftler stellen Rohdaten in Amazon Simple Storage Service (Amazon S3) zur Verfügung, führen explorative Datenanalysen (EDA) durch, entwickeln das erste ML-Modell und bewerten dessen Inferenzleistung. Sie können diese Aktivitäten interaktiv über Jupyter-Notebooks durchführen.
- Schritt 2. Erstellen Sie die Runtime-Skripten — Sie integrieren das Modell in Runtime-Python-Skripten, sodass es von einem ML-Framework (in unserem Fall Amazon SageMaker AI) verwaltet

und bereitgestellt werden kann. Dies ist der erste Schritt, um von der interaktiven Entwicklung eines eigenständigen Modells zur Produktion überzugehen. Insbesondere definieren Sie die Logik für die Vorverarbeitung, Bewertung, Schulung und Inferenz separat.

- Schritt 3. Definieren Sie die Pipeline — Sie definieren die Eingabe- und Ausgabe-Platzhalter für jeden Schritt der Pipeline. Konkrete Werte für diese werden später, während der Laufzeit, bereitgestellt (Schritt 5). Sie konzentrieren sich auf Pipelines für Training, Inferenz, Kreuzvalidierung und Backtesting.
- Schritt 4. Pipeline erstellen — Sie erstellen die zugrunde liegende Infrastruktur, einschließlich der AWS Step Functions State Machine-Instanz, automatisiert (fast mit einem Klick), indem AWS CloudFormation Sie.
- Schritt 5. Pipeline ausführen — Sie führen die in Schritt 4 definierte Pipeline aus. Außerdem bereiten Sie Metadaten und Daten oder Datenspeicherorte vor, um konkrete Werte für die input/output Platzhalter einzugeben, die Sie in Schritt 3 definiert haben. Dazu gehören die in Schritt 2 definierten Runtime-Skripten sowie Modell-Hyperparameter.
- Schritt 6. Erweiterung der Pipeline — Sie implementieren Prozesse für kontinuierliche Integration und kontinuierliche Bereitstellung (CI/CD), automatisierte Umschulungen, geplante Inferenzen und ähnliche Erweiterungen der Pipeline.

Das folgende Diagramm veranschaulicht die wichtigsten Schritte dieses Prozesses.



Steps for creating the training and inference pipeline

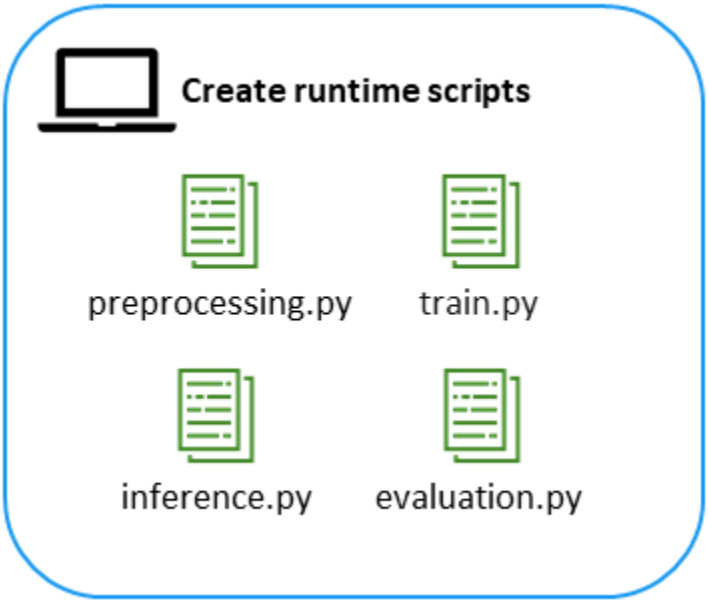
Schritt 1. Führen Sie EDA durch und entwickeln Sie das erste Modell

In diesem Schritt führen Datenwissenschaftler eine explorative Datenanalyse (EDA) durch, um den ML-Anwendungsfall und die Daten zu verstehen. Anschließend entwickeln sie die ML-Modelle (z. B. Klassifikations- und Regressionsmodelle), um das Problem in einem bestimmten Anwendungsfall zu lösen. Während der Modellentwicklung macht der Datenwissenschaftler häufig Annahmen über Inputs und Outputs, wie etwa Datenformate, den Datenlebenszyklus und die Orte der Zwischenausgabe. Diese Annahmen sollten dokumentiert werden, damit sie bei den Komponententests in Schritt 2 zur Überprüfung verwendet werden können.

Obwohl sich dieser Schritt auf die Modellentwicklung konzentriert, müssen Datenwissenschaftler oft eine Mindestmenge an Hilfscode für die Vorverarbeitung, Schulung, Bewertung und Inferenz schreiben. Der Datenwissenschaftler sollte in der Lage sein, diesen Code in der Entwicklungsumgebung auszuführen. Wir empfehlen außerdem, optionale Laufzeitargumente anzugeben, damit dieser Hilfscode dynamisch für die Ausführung in anderen Umgebungen ohne umfangreiche manuelle Änderungen konfiguriert werden kann. Dadurch wird die Integration zwischen dem Modell und der Pipeline in den Schritten 2 und 3 beschleunigt. Beispielsweise sollte Code zum Lesen der Rohdaten in Funktionen gekapselt werden, damit Daten konsistent vorverarbeitet werden können.

Wir empfehlen, mit einem Framework wie scikit-learn,, XGBoostPyTorch, Keras zu beginnen oder das ML-Modell und seinen TensorFlowHilfscode zu entwickeln. Scikit-Learn ist beispielsweise eine kostenlose ML-Bibliothek, die in Python geschrieben ist. Sie bietet eine einheitliche API-Konvention für Objekte und umfasst vier Hauptobjekte — Schätzer, Prädiktor, Transformator und Modell —, die einfache Datentransformationen abdecken, Label- und Feature-Engineering unterstützen und Vorverarbeitungs- und Modellierungsschritte kapseln. Diese Objekte tragen dazu bei, die Verbreitung von Standardcode zu verhindern und zu verhindern, dass Validierungs- und Testdaten in den Trainingsdatensatz gelangen. Ebenso verfügt jedes ML-Framework über eine eigene Implementierung der wichtigsten ML-Artefakte. Wir empfehlen Ihnen, bei der Entwicklung von ML-Modellen die API-Konventionen Ihres ausgewählten Frameworks einzuhalten.

Schritt 2. Erstellen Sie die Runtime-Skripten



In diesem Schritt integrieren Sie das Modell, das Sie in Schritt 1 entwickelt haben, und den zugehörigen Hilfscode in eine ML-Plattform für produktionsbereites Training und Inferenz. Konkret beinhaltet dies die Entwicklung von Runtime-Skripten, damit das Modell in KI integriert werden kann. SageMaker Diese eigenständigen Python-Skripte enthalten vordefinierte SageMaker KI-Callback-Funktionen und Umgebungsvariablen. Sie werden in einem SageMaker KI-Container ausgeführt, der auf einer Amazon Elastic Compute Cloud (Amazon EC2) -Instance gehostet wird. Die Amazon SageMaker AI Python SDK-Dokumentation enthält detaillierte Informationen darüber, wie diese Callbacks und die Hilfseinstellungen für Training und Inferenz zusammenarbeiten. Die folgenden Abschnitte enthalten zusätzliche Empfehlungen für die Entwicklung von ML-Runtime-Skripten, die auf unseren Erfahrungen in der Zusammenarbeit mit AWS Kunden basieren.

Verwendung von Verarbeitungsaufträgen

SageMaker KI bietet zwei Optionen für die Durchführung von Modellinferenzen im Batch-Modus. Sie können einen SageMaker KI-Verarbeitungsjob oder einen Batch-Transformationsjob verwenden. Jede Option hat Vor- und Nachteile.

Ein Verarbeitungsjob besteht aus einer Python-Datei, die in einem SageMaker AI-Container ausgeführt wird. Der Verarbeitungsjob besteht aus der Logik, die Sie in Ihre Python-Datei einfügen. Er hat folgende Vorteile:

- Wenn Sie die grundlegende Logik eines Schulungsjobs verstehen, sind Verarbeitungsjobs einfach einzurichten und leicht zu verstehen. Sie haben dieselben Abstraktionen wie Trainingsjobs (z. B. die Optimierung der Anzahl der Instanzen und der Datenverteilung).
- Datenwissenschaftler und ML-Ingenieure haben die volle Kontrolle über die Optionen zur Datenmanipulation.
- Der Datenwissenschaftler muss keine I/O Komponentenlogik verwalten, mit Ausnahme der vertrauten read/write Funktionen.
- Es ist etwas einfacher, die Dateien in Umgebungen ohne SageMaker KI auszuführen, was eine schnelle Entwicklung und lokale Tests unterstützt.
- Wenn ein Fehler auftritt, schlägt ein Verarbeitungsauftrag fehl, sobald das Skript fehlschlägt, und es wird nicht unerwartet auf einen erneuten Versuch gewartet.

Andererseits sind Batch-Transformationsjobs eine Erweiterung des Konzepts eines SageMaker KI-Endpunkts. Zur Laufzeit importieren diese Jobs Callback-Funktionen, die dann das Lesen der I/O Daten, das Laden des Modells und das Treffen der Vorhersagen übernehmen. Batch-Transformationsjobs haben folgende Vorteile:

- Sie verwenden eine Abstraktion der Datenverteilung, die sich von der Abstraktion unterscheidet, die bei Trainingsjobs verwendet wird.
- Sie verwenden dieselbe Kerndatei- und Funktionsstruktur sowohl für Batch-Inferenz als auch für Echtzeit-Inferenz, was praktisch ist.
- Sie verfügen über einen integrierten, auf Wiederholungsversuchen basierenden Fehlertoleranzmechanismus. Wenn beispielsweise bei einem Stapel von Datensätzen ein Fehler auftritt, wird der Vorgang mehrmals wiederholt, bevor der Job als Fehler beendet wird.

Aufgrund seiner Transparenz, seiner Benutzerfreundlichkeit in mehreren Umgebungen und seiner gemeinsamen Abstraktion mit Trainingsjobs haben wir uns in der Referenzarchitektur, die in diesem Handbuch vorgestellt wird, entschieden, den Verarbeitungsjob anstelle des Batch-Transformationsjobs zu verwenden.

Sie sollten Python-Runtime-Skripten lokal ausführen, bevor Sie sie in der Cloud bereitstellen. Insbesondere empfehlen wir, dass Sie die Main Guard-Klausel verwenden, wenn Sie Ihre Python-Skripte strukturieren und Unit-Tests durchführen.

Verwenden Sie die Main Guard-Klausel

Verwenden Sie eine Haupt-Guard-Klausel, um den Modulimport zu unterstützen und Ihr Python-Skript auszuführen. Das individuelle Ausführen von Python-Skripten ist für das Debuggen und Isolieren von Problemen in der ML-Pipeline von Vorteil. Wir empfehlen die folgenden Schritte:

- Verwenden Sie einen Argumentparser in den Python-Verarbeitungsdateien, um input/output Dateien und ihre Speicherorte anzugeben.
- Stellen Sie eine Hauptanleitung und Testfunktionen für jede Python-Datei bereit.
- Nachdem Sie eine Python-Datei getestet haben, integrieren Sie sie in die verschiedenen Phasen der ML-Pipeline, unabhängig davon, ob Sie ein AWS Step Functions Modell oder einen SageMaker KI-Verarbeitungsjob verwenden.
- Verwenden Sie Assert-Anweisungen in kritischen Abschnitten des Skripts, um das Testen und Debuggen zu erleichtern. Sie können beispielsweise eine Assert-Anweisung verwenden, um sicherzustellen, dass die Anzahl der Datensatz-Features nach dem Laden konsistent ist.

Komponententests

Unit-Tests von Runtime-Skripten, die für die Pipeline geschrieben wurden, sind eine wichtige Aufgabe, die bei der Entwicklung von ML-Pipelines häufig ignoriert wird. Dies liegt daran, dass maschinelles Lernen und Datenwissenschaft relativ neue Bereiche sind und etablierte Methoden der Softwareentwicklung wie Unit-Tests nur langsam eingeführt haben. Da die ML-Pipeline in der Produktionsumgebung verwendet wird, ist es wichtig, den Pipeline-Code zu testen, bevor das ML-Modell auf reale Anwendungen angewendet wird.

Unit-Tests des Runtime-Skripts bieten außerdem die folgenden einzigartigen Vorteile für ML-Modelle:

- Es verhindert unerwartete Datentransformationen. Die meisten ML-Pipelines beinhalten viele Datentransformationen. Daher ist es wichtig, dass diese Transformationen erwartungsgemäß funktionieren.
- Es bestätigt die Reproduzierbarkeit des Codes. Jede Zufälligkeit im Code kann durch Komponententests mit unterschiedlichen Anwendungsfällen erkannt werden.
- Es erzwingt die Modularität des Codes. Komponententests werden in der Regel mit dem Maß für die Testabdeckung verknüpft. Dabei handelt es sich um den Grad, in dem eine bestimmte Testsuite (eine Sammlung von Testfällen) den Quellcode eines Programms ausführt. Um eine hohe Testabdeckung zu erreichen, modularisieren Entwickler den Code, da es schwierig ist,

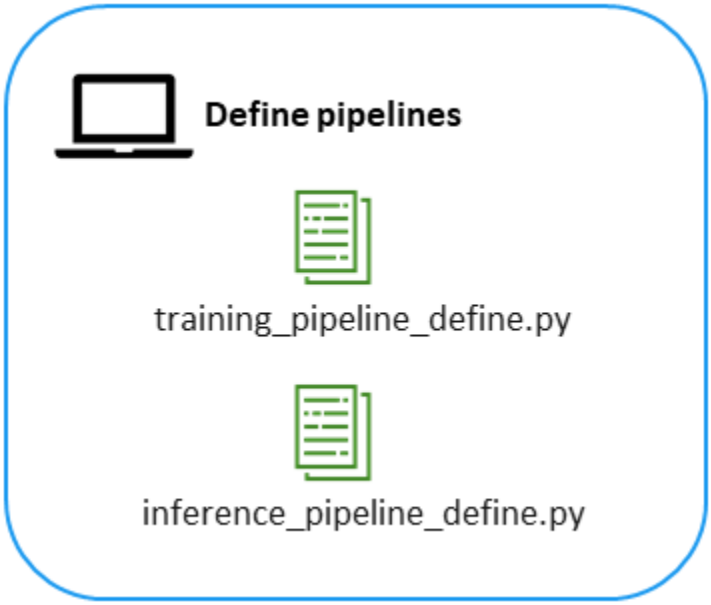
Komponententests für eine große Menge Code zu schreiben, ohne ihn in Funktionen oder Klassen zu unterteilen.

- Dadurch wird verhindert, dass minderwertiger Code oder Fehler in die Produktion einfließen.

Wir empfehlen Ihnen, ein ausgereiftes Unit-Test-Framework wie pytest zu verwenden, um die Unit-Test-Fälle zu schreiben, da es einfacher ist, umfangreiche Unit-Tests innerhalb eines Frameworks zu verwalten.

> **Important**
>
> Unit-Tests können nicht garantieren, dass alle Eckfälle getestet werden, aber sie können Ihnen helfen, Fehler proaktiv zu vermeiden, bevor Sie das Modell bereitstellen. Wir empfehlen, das Modell auch nach der Implementierung zu überwachen, um eine optimale Betriebsführung sicherzustellen.

Schritt 3. Definieren Sie die Pipeline



In diesem Schritt werden die Reihenfolge und Logik der Aktionen definiert, die die Pipeline ausführen wird. Dies umfasst diskrete Schritte sowie deren logische Ein- und Ausgänge. Wie ist beispielsweise der Status der Daten zu Beginn der Pipeline? Stammen sie aus mehreren Dateien mit unterschiedlichen Granularitätsebenen oder aus einer einzigen Flat-Datei? Wenn die Daten aus mehreren Dateien stammen, benötigen Sie einen einzigen Schritt für alle Dateien oder separate Schritte für jede Datei, um die Vorverarbeitungslogik zu definieren? Die Entscheidung hängt von der Komplexität der Datenquellen und dem Umfang ab, in dem sie vorverarbeitet werden.

In unserer Referenzimplementierung verwenden wir AWS Step Functions, einen serverlosen Funktions-Orchestrator, um die Workflow-Schritte zu definieren. Das ML Max-Framework unterstützt jedoch auch andere Pipeline- oder Zustandsmaschinensysteme wie Apache AirFlow (siehe Abschnitt Verschiedene Engines für die Pipeline-Orchestrierung), um die Entwicklung und Bereitstellung von ML-Pipelines voranzutreiben.

Verwenden des Step Functions SDK

Um die ML-Pipeline zu definieren, verwenden wir zunächst die vom AWS Step Functions Data Science SDK (dem Step Functions SDK) bereitgestellte Python-API auf hoher Ebene, um zwei Schlüsselkomponenten der Pipeline zu definieren: Schritte und Daten. Wenn Sie sich eine Pipeline als einen gerichteten azyklischen Graphen (DAG) vorstellen, stellen Schritte die Knoten im Graphen

dar, und Daten werden als gerichtete Kanten dargestellt, die einen Knoten (Schritt) mit dem nächsten verbinden. Zu den typischen Beispielen für ML-Schritte gehören die Vorverarbeitung, das Training und die Auswertung. Das Step Functions SDK bietet eine Reihe von integrierten Schritten (z. B. die TrainingStep), die Sie verwenden können. Beispiele für Daten sind Eingabe, Ausgabe und viele Zwischendatensätze, die durch einige Schritte in der Pipeline erzeugt werden. Wenn Sie eine ML-Pipeline entwerfen, kennen Sie die konkreten Werte der Datenelemente nicht. Sie können Datenplatzhalter definieren, die als Vorlage dienen (ähnlich wie Funktionsparameter) und nur den Namen des Datenelements und primitive Datentypen enthalten. Auf diese Weise können Sie einen vollständigen Pipeline-Blueprint entwerfen, ohne die konkreten Werte der Daten, die sich auf dem Diagramm bewegen, im Voraus zu kennen. Zu diesem Zweck können Sie die Platzhalterklassen im Step Functions SDK verwenden, um diese Datenvorlagen explizit zu modellieren.

ML-Pipelines benötigen außerdem Konfigurationsparameter, um das Verhalten der einzelnen ML-Schritte detailliert steuern zu können. Diese speziellen Datenplatzhalter werden als Parameterplatzhalter bezeichnet. Viele ihrer Werte sind bei der Definition der Pipeline unbekannt. Zu den Platzhaltern für Parameter gehören beispielsweise infrastrukturbezogene Parameter, die Sie beim Pipeline-Design definieren (z. B. die URL eines Container-Images), AWS-Region und Parameter im Zusammenhang mit der ML-Modellierung (wie Hyperparameter), die Sie bei der Ausführung der Pipeline definieren.

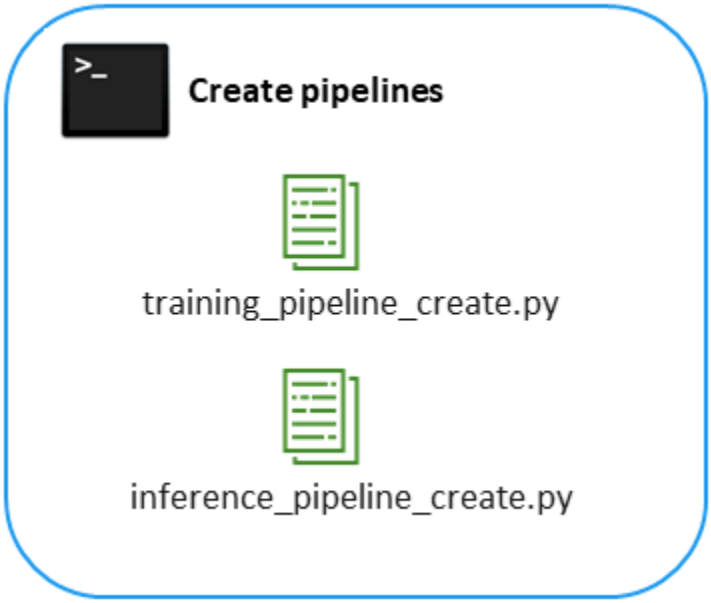
Erweiterung des Step Functions SDK

In unserer Referenzimplementierung bestand eine Anforderung darin, ML-Pipeline-Definitionen mithilfe spezifischer Parametereinstellungen von der konkreten Erstellung und Bereitstellung von ML-Pipelines zu trennen. Einige der integrierten Schritte im Step Functions SDK erlaubten es uns jedoch nicht, all diese Platzhalterparameter zu übergeben. Stattdessen wurde erwartet, dass die Parameterwerte direkt während der Pipeline-Entwurfszeit durch API-Aufrufe für die SageMaker KI-Konfiguration abgerufen werden. Dies funktioniert einwandfrei, wenn die SageMaker KI-Entwurfszeitumgebung mit der SageMaker KI-Laufzeitumgebung identisch ist. In realen Umgebungen ist dies jedoch selten der Fall. Eine solch enge Kopplung zwischen der Entwurfszeit der Pipeline und der Laufzeit sowie die Annahme, dass die ML-Plattforminfrastruktur konstant bleibt, behindern die Anwendbarkeit der entworfenen Pipeline erheblich. Tatsächlich bricht die ML-Pipeline sofort ab, wenn die zugrunde liegende Bereitstellungsplattform auch nur die geringste Änderung erfährt.

Um diese Herausforderung zu bewältigen und eine robuste ML-Pipeline zu erstellen (die wir einmal entwerfen und überall ausführen wollten), haben wir unsere eigenen benutzerdefinierten Schritte implementiert, indem wir einige der integrierten Schritte erweitert haben, darunter

TrainingStepModelStep, und. TransformerStep Diese Erweiterungen werden im ML Max-Projekt bereitgestellt. Die Schnittstellen dieser benutzerdefinierten Schritte unterstützen weitaus mehr Platzhalter für Parameter, die während der Pipelineerstellung oder während des Betriebs der Pipeline mit konkreten Werten gefüllt werden können.

Schritt 4. Erstellen Sie die Pipeline



Nachdem Sie die Pipeline logisch definiert haben, ist es an der Zeit, die Infrastruktur zur Unterstützung der Pipeline zu erstellen. Für diesen Schritt sind mindestens die folgenden Funktionen erforderlich:

- Speicher zum Hosten und Verwalten von Pipeline-Eingaben und -Ausgaben, einschließlich Code, Modellartefakten und Daten, die in Trainings- und Inferenzläufen verwendet werden.
- Rechenleistung (GPU oder CPU) für Modellierung und Inferenz sowie für die Vor- und Nachverarbeitung von Daten.
- Orchestrierung, um die verwendeten Ressourcen zu verwalten und alle regelmäßigen Läufe zu planen. Beispielsweise kann das Modell in regelmäßigen Abständen neu trainiert werden, sobald neue Daten verfügbar werden.
- Protokollierung und Warnmeldungen zur Überwachung der Genauigkeit des Pipeline-Modells, zur Ressourcennutzung und zur Fehlerbehebung.

Implementierung mit AWS CloudFormation

Um die von uns verwendete Pipeline zu erstellen AWS CloudFormation, bei der es sich um einen AWS Dienst für die Bereitstellung und Verwaltung der Infrastruktur als Code handelt. Die AWS CloudFormation Vorlagen enthalten die Step Functions-Definition, die im vorherigen Schritt mit

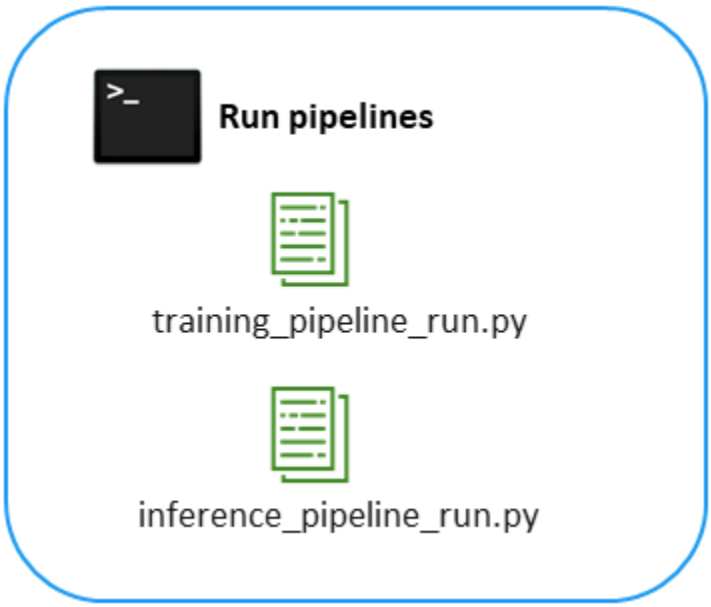
dem Step Functions SDK erstellt wurde. Dieser Schritt beinhaltet die Erstellung der Step Functions-Instanz, die als AWS-managed Step Functions Functions-Zustandsmaschine bezeichnet wird. In dieser Phase werden keine Ressourcen für Training und Inferenz geschaffen, da Schulungs- und Inferenzjobs als SageMaker KI-Jobs bei Bedarf und nur dann ausgeführt werden, wenn sie benötigt werden. Dieser Schritt umfasst auch das Erstellen von AWS Identity and Access Management (IAM-) Rollen zum Ausführen der Step Functions, zum Ausführen von SageMaker KI sowie zum Lesen und Schreiben von Amazon S3.

Änderung der Ausgabe aus dem Step Functions SDK

Wir mussten einige geringfügige Änderungen an der CloudFormation Ausgabe aus dem vorherigen Abschnitt vornehmen. Wir haben den einfachen Python-String-Matching verwendet, um Folgendes zu tun:

- Wir haben Logik für die Erstellung des `Parameters` Abschnitts der CloudFormation Vorlage hinzugefügt. Dies liegt daran, dass wir zwei Rollen erstellen und den Pipeline-Namen zusammen mit der Bereitstellungsumgebung als Parameter definieren möchten. Dieser Schritt umfasst auch alle zusätzlichen Ressourcen und Rollen, die Sie möglicherweise erstellen möchten, wie in Schritt 6 beschrieben.
- Wir haben drei Felder neu formatiert, sodass sie das erforderliche `!Sub` Präfix und die erforderlichen Anführungszeichen haben, sodass sie im Rahmen des Bereitstellungsprozesses dynamisch aktualisiert werden können:
  - Die `StateMachineName` Eigenschaft, die den Zustandsmaschine benennt.
  - Die `DefinitionString` Eigenschaft, die den Zustandsmaschine definiert.
  - Die `RoleArn` Eigenschaft, die von der Zustandsmaschine zurückgegeben wird.

Schritt 5. Führen Sie die Pipeline aus



In diesem Schritt wird die Trainings- oder Inferenzpipeline ausgeführt, die in Schritt 4 in den CloudFormation Stacks erstellt wurde. Die Pipeline kann erst ausgeführt werden, wenn ihre internen Platzhalterparameter mit konkreten Werten gefüllt wurden. Diese Aktion des Zuweisens von Werten zu Platzhalterparametern ist die Hauptaktivität von Schritt 5. Zu den Platzhalterparametern gehören beispielsweise:

- Der Speicherort von Eingabe-, Ausgabe- und Zwischendatensätzen
- Der Amazon S3 S3-Speicherort der Runtime-Skripts und anderer Vorverarbeitungs- oder Evaluierungscodes, die in Schritt 2 entwickelt wurden (z. B. `sm_submit_url` für die Trainingspipeline)
- Der Name der Region AWS

Sie müssen sicherstellen, dass diese Pfadwerte auf gültige Daten oder Code verweisen, bevor Sie die Pipeline ausführen. Wenn Sie beispielsweise den Platzhalterparameter auffüllen, der die Amazon S3 S3-URL der Python-Runtime-Skripten darstellt, müssen Sie diese Skripten an diese URL hochladen. Die Person, die die Pipeline betreibt, ist für die Konsistenzprüfung und das Hochladen von Daten verantwortlich. Personen, die die Pipeline definieren oder erstellen, müssen sich darüber keine Gedanken machen.

Je nach Reifegrad der Pipeline kann dieser Schritt automatisiert werden, sodass er regelmäßig (wöchentlich oder monatlich) ausgeführt wird. Die Automatisierung erfordert auch eine solide Überwachung, was ein wichtiger Bereich ist, der jedoch nicht in den Geltungsbereich dieses Leitfadens fällt. Für die Durchführung der Schulungspipeline wäre es angemessen, die Bewertungskennzahlen zu überwachen. Für die Inferenz-Pipeline wäre es angemessen, die Drift bei der Verteilung der Eingabedaten zu überwachen und, wenn möglich, regelmäßig Kennzeichnungen zu sammeln und die Abweichung bei der Vorhersagegenauigkeit zu messen. Diese Aufzeichnungen aus den Trainings- und Inferenzläufen sollten in einer Datenbank aufgezeichnet werden, damit sie zu einem späteren Zeitpunkt analysiert werden können.

Schritt 6: Erweitern Sie die Pipeline

In diesem Leitfaden wird erklärt, wie Sie AWS schnell mit dem Aufbau von ML-Pipelines beginnen können, und zwar mit konkreter Architektur. Für die Weiterentwicklung der Pipeline sind weitere Überlegungen erforderlich, wie z. B. die Verwaltung von Metadaten, die Nachverfolgung von Experimenten und die Überwachung. Dies sind wichtige Themen, die nicht in den Geltungsbereich dieses Leitfadens fallen. In den folgenden Abschnitten wird ein weiterer Aspekt des Pipeline-Managements behandelt, nämlich die Pipeline-Automatisierung.

Verschiedene Automatisierungsstufen

Sie können eine Trainingspipeline zwar manuell in der SageMaker KI-Konsole einrichten, in der Praxis empfehlen wir jedoch, manuelle Berührungspunkte bei der Bereitstellung von ML-Trainingspipelines zu minimieren, um sicherzustellen, dass ML-Modelle konsistent und wiederholt eingesetzt werden. Abhängig von Ihren Anforderungen und den Geschäftsproblemen, mit denen Sie sich befassen, können Sie eine Implementierungsstrategie auf drei Ebenen festlegen und umsetzen: halbautomatisiert, vollautomatisch und vollständig verwaltet.

- Halbautomatisiert — Die im vorherigen Abschnitt erläuterten Schritte folgen standardmäßig einem halbautomatischen Ansatz, da sie die Trainings- und Inferenz-Pipeline mithilfe von Vorlagen bereitstellen. CloudFormation Dadurch wird die Reproduzierbarkeit der Pipeline gewährleistet und Sie können sie problemlos ändern und aktualisieren.
- Vollständig automatisiert — Eine fortgeschrittenere Option ist die Verwendung von kontinuierlicher Integration und kontinuierlicher Bereitstellung (durch CI/CD) to the development, staging, and production environments. Incorporating CI/CD Methoden zur Bereitstellung der Schulungspipeline kann sichergestellt werden, dass die Automatisierung sowohl Rückverfolgbarkeit als auch Qualitätskontrollen umfasst).
- Vollständig verwaltet — Letztlich können Sie ein vollständig verwaltetes System entwickeln, sodass Sie eine ML-Trainingspipeline mit einer Reihe einfacher Manifeste bereitstellen können und das System die erforderlichen AWS Dienste selbst konfigurieren und koordinieren kann.

In diesem Leitfaden haben wir uns entschieden, eine konkrete Architektur vorzustellen. Es gibt jedoch alternative Technologien, die Sie in Betracht ziehen können. In den nächsten beiden Abschnitten werden einige alternative Optionen für die Plattform und die Orchestrierungs-Engine erörtert.

Verschiedene Plattformen für ML-Workloads

Amazon SageMaker AI ist der AWS verwaltete Service für das Training und die Bereitstellung von ML-Modellen. Viele Benutzer schätzen die Vielzahl der integrierten Funktionen und die vielen Optionen, die es für die Ausführung von ML-Workloads bietet. SageMaker KI ist besonders nützlich, wenn Sie gerade erst mit der Implementierung von ML in der Cloud beginnen. Zu den wichtigsten Funktionen von SageMaker KI gehören:

- Integrierte Rückverfolgbarkeit (einschließlich Kennzeichnung, Schulung, Modellverfolgung, Optimierung und Inferenz).
- Integrierte Ein-Klick-Optionen für Training und Inferenz mit minimaler Python- und ML-Erfahrung.
- Erweitertes Hyperparameter-Tuning.
- Support für alle wichtigen Frameworks für künstliche Intelligenz und maschinelles Lernen (ML/KI) sowie für benutzerdefinierte Docker-Container.
- Integrierte Überwachungsfunktionen.
- Integrierte Nachverfolgung von Historien, einschließlich Trainingsaufträgen, Verarbeitungsaufträgen, Batch-Transformationsaufträgen, Modellen, Endpunkten und Durchsuchbarkeit. Einige Historien, wie z. B. Training, Verarbeitung und Batch-Transformation, sind unveränderlich und können nur angehängt werden.

Eine der Alternativen zur Verwendung von KI ist. SageMaker AWS Batch AWS Batch bietet ein geringeres Maß an Kontrolle über die Berechnung und Orchestrierung für Ihre Umgebung, ist aber nicht speziell für maschinelles Lernen konzipiert. Zu seinen wichtigsten Funktionen gehören:

- Out-of-the-box automatische Skalierung der Rechenressourcen auf der Grundlage der Arbeitslast.
- Out-of-the-box Unterstützung für Auftragspriorität, Wiederholungsversuche und Aufgabenabhängigkeiten.
- Warteschlangenbasierter Ansatz, der die Erstellung von wiederkehrenden Aufträgen und Aufträgen auf Abruf unterstützt.
- Support für CPU- und GPU-Workloads. Die Fähigkeit, GPU für die Erstellung von ML-Modellen zu verwenden, ist von entscheidender Bedeutung, da die GPU den Trainingsprozess erheblich beschleunigen kann, insbesondere bei Deep-Learning-Modellen.
- Möglichkeit, ein benutzerdefiniertes Amazon Machine Image (AMI) für die Rechenumgebung zu definieren.

Verschiedene Engines für die Pipeline-Orchestrierung

Die zweite Hauptkomponente ist die Pipeline-Orchestrierungsschicht. AWS bietet Step Functions für eine vollständig verwaltete Orchestrierungserfahrung. Eine beliebte Alternative zu Step Functions ist Apache Airflow. Wenn Sie eine Entscheidung zwischen den beiden treffen, sollten Sie Folgendes berücksichtigen:

- Erforderliche Infrastruktur — AWS Step Functions ist ein vollständig verwalteter Service, der serverlos ist, wohingegen Airflow die Verwaltung Ihrer eigenen Infrastruktur erfordert und auf Open-Source-Software basiert. Dadurch bietet Step Functions sofort eine hohe Verfügbarkeit, wohingegen die Verwaltung von Apache Airflow zusätzliche Schritte erfordert.
- Planungsfunktionen — Sowohl Step Functions als auch Airflow bieten vergleichbare Funktionen.
- Visualisierungsmöglichkeiten und Benutzeroberfläche — Sowohl Step Functions als auch Airflow bieten vergleichbare Funktionen.
- Übergabe von Variablen innerhalb des Berechnungsdiagramms — Step Functions bietet eingeschränkte Funktionen für die Verwendung von AWS Lambda Funktionen, wohingegen Airflow Schnittstellen bereitstellt XCom .
- Verwendung — Step Functions ist bei AWS Kunden sehr beliebt, und Airflow wurde von der Datentechnik-Community weitgehend übernommen.

Schlussfolgerung

Im Zuge des Übergangs von maschinellem Lernen von einer Forschungsdisziplin zu einem angewandten Bereich haben wir in verschiedenen Branchen ein jährliches Wachstum von 25 Prozent bei der Entwicklung, dem Einsatz und dem Betrieb von ML-Pipelines verzeichnet. Der Geschäftswert von ML wird durch day-to-day ML-Operationen und -Pipelines realisiert, die wiederum die Erforschung und Entwicklung von ML-Modellen und -Algorithmen vorantreiben. Dennoch bringt der Einsatz von ML in der Produktion zahlreiche Herausforderungen mit sich, da er erheblich unterschiedliche Aktivitäten und Artefakte wie Datenmanagement, Verarbeitung, Analyse, Modellierung, Verifizierung und Sicherheit miteinander verknüpft. In zahlreichen KI/ML-Projekten mit AWS Kunden hat unser Data-Science-Team festgestellt, dass eine zentrale Herausforderung das Fehlen eines end-to-end Workflows ist, der eine Reihe von Vorlagen für die optimale Verschmelzung oder Trennung verschiedener ML-Aktivitäten und -Artefakte bereitstellen würde. DevOps In diesem Leitfaden haben wir den ML Max-Workflow vorgestellt, um dieses dringende Problem zu lösen. ML Max bietet step-by-step Richtlinien und eine Reihe von Programmiervorlagen. Ziel ist es, einen schnellen und kostengünstigen Übergang von einer interaktiven Modellentwicklungsphase zu einer vollständigen, skalierbaren ML-Pipeline-Konfiguration zu ermöglichen, die für die Produktion bereit ist.

Weitere Ressourcen

Verwandte Anleitungen und Muster

- Quantifizierung der Unsicherheit in Deep-Learning-Systemen
- Migrieren Sie mithilfe von AWS Entwicklertools Workloads zum Erstellen, Trainieren und Bereitstellen von ML-Workloads zu Amazon SageMaker AI
- AWS Website mit präskriptiven Leitlinien

AWS Dienstleistungen

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

AWS allgemeine Ressourcen

- AWS Dokumentation
- AWS allgemeine Referenz
- AWS Glossar

Dokumentverlauf

In der folgenden Tabelle werden wichtige Änderungen in diesem Leitfaden beschrieben. Um Benachrichtigungen über zukünftige Aktualisierungen zu erhalten, können Sie einen RSS-Feed abonnieren.

Änderung	Beschreibung	Datum
Erste Veröffentlichung	—	29. Januar 2021

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.