Criação de pipelines de ML prontos para produção em AWS

AWS Recomendações

AWS Recomendações: Criação de pipelines de ML prontos para produção em AWS

Table of Contents

Criação de pipelines de ML prontos para produção em AWS

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

Janeiro de 2021 (histórico do documento)

Os projetos de machine learning (ML) exigem um esforço significativo em vários estágios que inclui modelagem, implantação e produção para agregar valor comercial e resolver problemas do mundo real. Várias alternativas e opções de personalização estão disponíveis em cada etapa, o que torna cada vez mais difícil preparar um modelo de ML para produção dentro das restrições de seus recursos e orçamento. Nos últimos anos na Amazon Web Services (AWS), nossa equipe de Ciência de Dados trabalhou com diferentes setores da indústria em iniciativas de ML. Identificamos pontos problemáticos compartilhados por muitos AWS clientes, que se originam tanto de problemas organizacionais quanto de desafios técnicos, e desenvolvemos uma abordagem ideal para fornecer soluções de ML prontas para produção.

Este guia destina-se a cientistas de dados e engenheiros de ML envolvidos na implantação de pipelines de ML. Ele descreve nossa abordagem para o fornecimento de pipelines de ML prontos para produção. O guia discute como você pode fazer a transição da execução interativa de modelos de ML (durante o desenvolvimento) para sua implantação como parte de um pipeline (durante a produção) para seu caso de uso de ML. Para isso, também desenvolvemos um conjunto de modelos de exemplo (veja o projeto do projeto ML Max), para acelerar a entrega de soluções personalizadas de ML para produção, de modo que você possa começar a usar rapidamente sem precisar fazer muitas escolhas de design.

Visão geral do

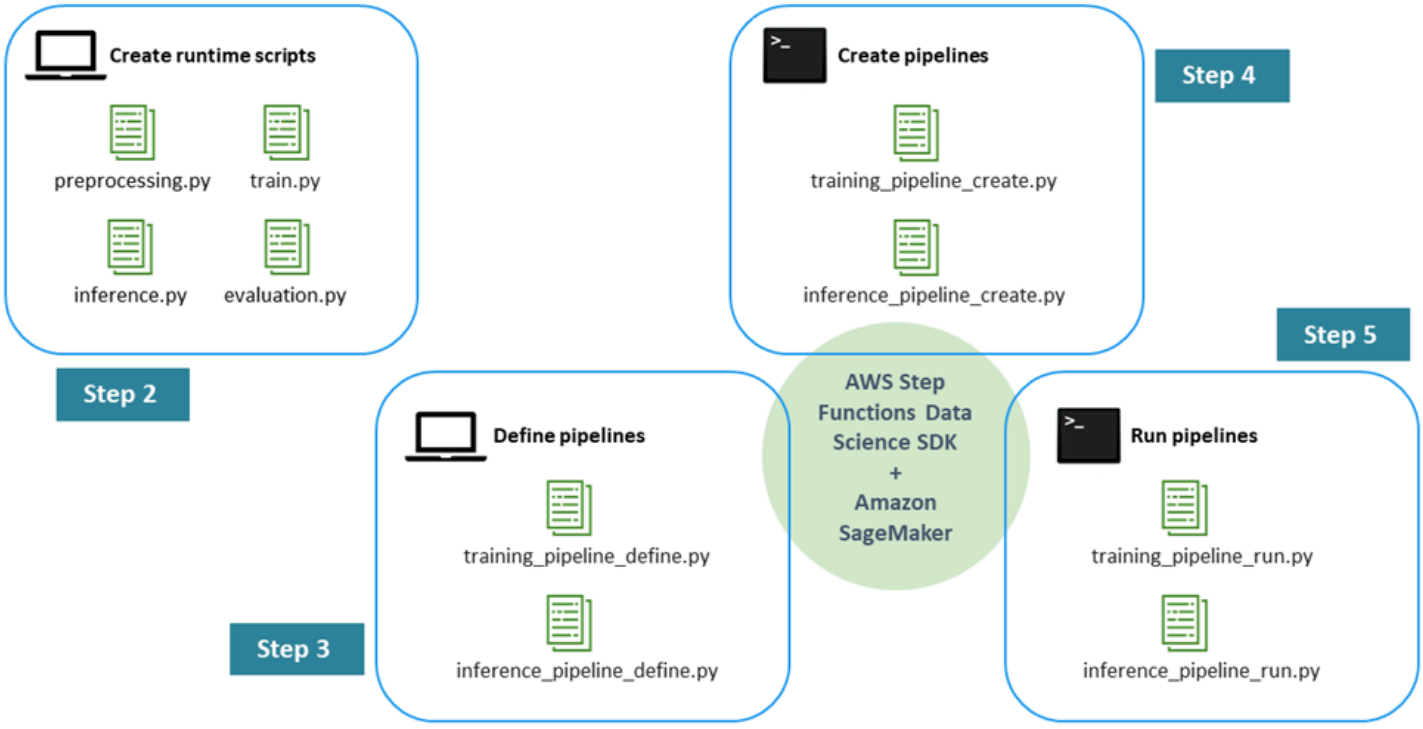
O processo de criação de um pipeline de ML pronto para produção consiste nas seguintes etapas:

- Etapa 1. Execute a EDA e desenvolva o modelo inicial — Cientistas de dados disponibilizam dados brutos no Amazon Simple Storage Service (Amazon S3), realizam análise exploratória de dados (EDA), desenvolvem o modelo de ML inicial e avaliam sua performance de inferência. Você pode realizar essas atividades de forma interativa por meio dos cadernos Jupyter.
- Etapa 2. Crie os scripts de tempo de execução — Você integra o modelo aos scripts Python de tempo de execução para que ele possa ser gerenciado e provisionado por uma estrutura de ML (no nosso caso, Amazon AI). SageMaker Esse é o primeiro passo para passar do

desenvolvimento interativo de um modelo independente para a produção. Especificamente, você define separadamente a lógica para pré-processamento, avaliação, treinamento e inferência.

- Etapa 3. Defina o pipeline — Você define os espaços reservados de entrada e saída para cada etapa do pipeline. Valores concretos para eles serão fornecidos mais tarde, durante o runtime (etapa 5). Você se concentra em pipelines para treinamento, inferência, validação cruzada e backtesting.
- Etapa 4. Crie o pipeline — Você cria a infraestrutura subjacente, incluindo a instância da máquina de AWS Step Functions estado de forma automatizada (quase um clique), usando AWS CloudFormation.
- Etapa 5. Execute o pipeline — Você executa o pipeline definido na etapa 4. Você também prepara metadados e dados ou locais de dados para preencher valores concretos para os input/output espaços reservados que você definiu na etapa 3. Isso inclui os scripts de runtime definidos na etapa 2, bem como os hiperparâmetros do modelo.
- Etapa 6. Expanda o pipeline — Você implementa processos de integração contínua e implantação contínua (CI/CD), reciclagem automatizada, inferência programada e extensões similares do pipeline.

O diagrama a seguir ilustra as principais etapas desse processo.



Steps for creating the training and inference pipeline

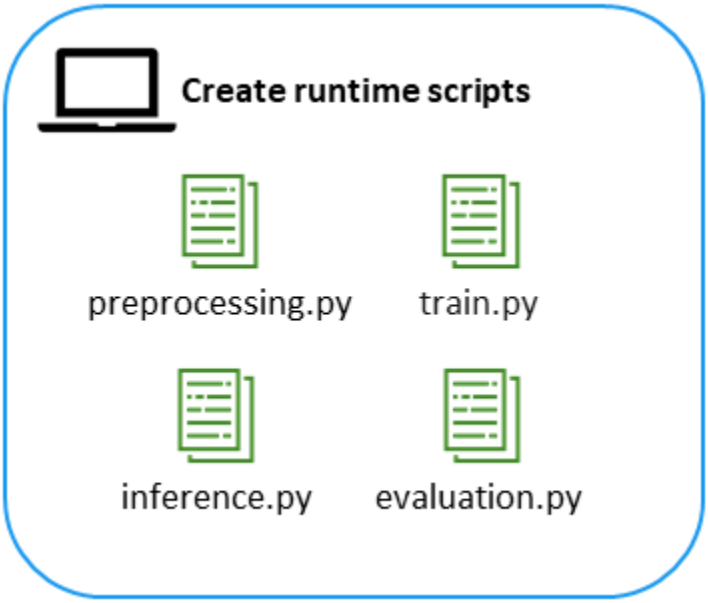
Etapa 1. Execute o EDA e desenvolva o modelo inicial

Nesta etapa, cientistas de dados realizam a análise exploratória de dados (EDA) para entender os dados e o caso de uso de ML. Em seguida, eles desenvolvem os modelos de ML (por exemplo, modelos de classificação e regressão) para resolver o problema em um determinado caso de uso. Durante o desenvolvimento do modelo, o cientista de dados geralmente faz suposições sobre entradas e saídas, como formatos de dados, ciclo de vida dos dados e locais de saída intermediária. Essas suposições devem ser documentadas para que possam ser usadas para verificação durante os testes de unidade na etapa 2.

Embora essa etapa se concentre no desenvolvimento de modelos, os cientistas de dados geralmente precisam escrever uma quantidade mínima de código auxiliar para pré-processamento, treinamento, avaliação e inferência. O cientista de dados deve ser capaz de executar esse código no ambiente de desenvolvimento. Também recomendamos fornecer argumentos de runtime opcionais para que esse código auxiliar possa ser configurado dinamicamente para ser executado em outros ambientes sem grandes alterações manuais. Isso irá acelera a integração entre o modelo e o pipeline nas etapas 2 e 3. Por exemplo, o código para ler os dados brutos deve ser encapsulado em funções para que os dados possam ser pré-processados de maneira consistente.

Recomendamos que você comece com uma estrutura como scikit-learn, XGBoost, PyTorch, Keras ou TensorFlow para desenvolver o modelo de ML e seu código auxiliar. Por exemplo, o scikit-learn é uma biblioteca de ML gratuita e escrita em Python. Ele fornece uma convenção de API uniforme para objetos e inclui quatro objetos principais — estimador, preditor, transformador e modelo — que abrangem transformações leves nos dados, oferecem suporte à engenharia de atributos e rótulos e encapsulam as etapas de pré-processamento e modelagem. Esses objetos ajudam a evitar a proliferação de códigos clichê e evitam que dados de validação e teste vazem para o conjunto de dados de treinamento. Da mesma forma, cada estrutura de ML tem sua própria implementação dos principais artefatos de ML e recomendamos que você cumpra as convenções de API da estrutura selecionada ao desenvolver modelos de ML.

Etapa 2. Crie os scripts de runtime



Nesta etapa, você integra o modelo desenvolvido na etapa 1 e seu código auxiliar associado em uma plataforma de ML para treinamento e inferência prontos para produção. Especificamente, isso envolve o desenvolvimento de scripts de tempo de execução para que o modelo possa ser incorporado à SageMaker IA. Esses scripts Python autônomos incluem funções de retorno de chamada de IA SageMaker predefinidas e variáveis de ambiente. Eles são executados dentro de um contêiner de SageMaker IA hospedado em uma instância do Amazon Elastic Compute Cloud (Amazon EC2). A documentação do Amazon SageMaker AI Python SDK fornece informações detalhadas sobre como esses retornos de chamada e a configuração auxiliar funcionam juntos para treinamento e inferência. As seções a seguir fornecem recomendações adicionais para o desenvolvimento de scripts de runtime de ML, com base em nossa experiência de trabalho com clientes AWS .

Como utilizar as tarefas de processamento

SageMaker A IA fornece duas opções para realizar a inferência do modelo em lote. Você pode usar um trabalho de processamento de SageMaker IA ou um trabalho de transformação em lote. Cada opção tem vantagens e desvantagens.

Um trabalho de processamento consiste em um arquivo Python executado dentro de um contêiner de SageMaker IA. A tarefa de processamento consiste em qualquer lógica que você coloque em seu arquivo Python. Ela tem estas vantagens:

- Quando você entender a lógica básica de uma tarefa de treinamento, as tarefas de processamento serão simples de configurar e fáceis de compreender. Elas compartilham as mesmas abstrações das tarefas de treinamento (por exemplo, ajustar a contagem de instâncias e a distribuição de dados).
- Cientistas de dados e engenheiros de ML possuem controle total sobre as opções de manipulação de dados.
- O cientista de dados não precisa gerenciar nenhuma lógica de I/O componentes, exceto a read/write funcionalidade familiar.
- É um pouco mais fácil executar os arquivos em ambientes sem SageMaker IA, o que ajuda no rápido desenvolvimento e nos testes locais.
- Se houver um erro, uma tarefa de processamento falhará assim que o script falhar e não haverá espera inesperada por uma nova tentativa.

Por outro lado, as tarefas de transformação em lote são uma extensão do conceito de endpoint de SageMaker IA. Em tempo de execução, esses trabalhos importam funções de retorno de chamada, que então lidam com a I/O leitura dos dados, o carregamento do modelo e a realização das previsões. As tarefas de transformação em lote possuem as seguintes vantagens:

- Elas utilizam uma abstração de distribuição de dados que difere da abstração utilizada em tarefas de treinamento.
- Elas utilizam o mesmo arquivo principal e a mesma estrutura de funções para inferência em lote e inferência em tempo real, o que é conveniente.
- Elas possuem um mecanismo de tolerância a falhas integrado baseado em novas tentativas. Por exemplo, se ocorrer um erro em um lote de registros, ela tentará novamente várias vezes antes que o trabalho seja encerrado com falha.

Por causa de sua transparência, facilidade de uso em vários ambientes e abstração compartilhada com tarefas de treinamento, decidimos utilizar a tarefa de processamento em vez da tarefa de transformação em lote na arquitetura de referência apresentada neste guia.

Você deve executar scripts de runtime do Python localmente antes de implantá-los na nuvem. Especificamente, recomendamos que você utilize a cláusula de proteção principal ao estruturar seus scripts Python e realizar testes de unidade.

Como utilizar a cláusula de proteção principal

Use uma cláusula de proteção principal para suporte à importação de módulos e executar seu script Python. Executar scripts Python individualmente é benéfico para depurar e isolar problemas no pipeline de ML. Recomendamos as seguintes etapas:

- Use um analisador de argumentos nos arquivos de processamento do Python para input/output especificar os arquivos e suas localizações.
- Forneça um guia principal e funções de teste para cada arquivo Python.
- Depois de testar um arquivo Python, incorpore-o aos diferentes estágios do pipeline de ML, esteja você usando um AWS Step Functions modelo ou uma tarefa de processamento de SageMaker IA.
- Use instruções Assert em seções críticas do script para facilitar o teste e a depuração. Por exemplo, você pode utilizar uma instrução Assert para garantir que o número de atributos do conjunto de dados seja consistente após o carregamento.

Teste de unidade

O teste de unidade de scripts de runtime que foram escritos para o pipeline é uma tarefa importante e frequentemente ignorada no desenvolvimento do pipeline de ML. Isso ocorre porque machine learning e ciência de dados são campos relativamente novos e demoraram a adotar práticas de engenharia de software bem estabelecidas, como testes de unidade. Como o pipeline de ML será utilizado no ambiente de produção, é essencial testar o código do pipeline antes de aplicar o modelo de ML a aplicativos reais.

O teste de unidade do script de runtime também oferece os seguintes benefícios exclusivos para modelos de ML:

- Ele evita transformações inesperadas de dados. A maioria dos pipelines de ML envolve muitas transformações de dados, por isso é fundamental que essas transformações funcionem conforme o esperado.
- Ele valida a reprodutibilidade do código. Qualquer aleatoriedade no código pode ser detectada por testes de unidade com diferentes casos de uso.
- Ele reforça a modularidade do código. Os testes de unidade geralmente são associados à medida de cobertura do teste, que é o grau em que uma suíte de testes específica (uma coleção de casos de teste) executa o código-fonte de um programa. Para obter uma alta cobertura de testes, os

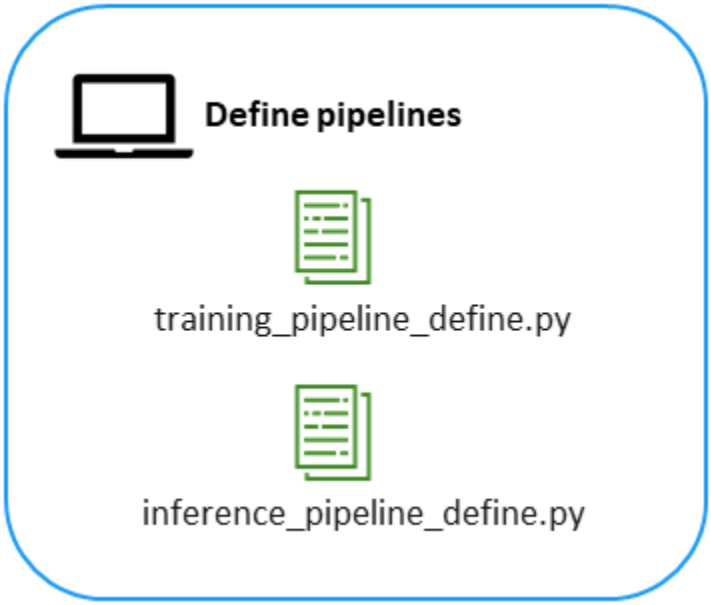
desenvolvedores modularizam o código, porque é difícil escrever testes de unidade para uma grande quantidade de código sem dividi-lo em funções ou classes.

- Isso evita que códigos ou erros de baixa qualidade sejam introduzidos na produção.

Recomendamos que você utilize uma estrutura de teste de unidade madura, como pytest, para escrever os casos de teste de unidade, porque é mais fácil gerenciar testes de unidade extensivos dentro de uma estrutura.

> **Important**
>
> Teste de unidade não garante que todos os casos extremos sejam testados, mas podem ajudá-lo a evitar erros de forma proativa antes de implantar o modelo. Recomendamos que você também monitore o modelo após a implantação para garantir a excelência operacional.

Etapa 3. Definir o pipeline



Nesta etapa, a sequência e a lógica das ações que o pipeline executará são definidas. Isso inclui etapas discretas, bem como suas entradas e saídas lógicas. Por exemplo, qual é o estado dos dados no início do pipeline? Eles vêm de vários arquivos com diferentes níveis de granularidade ou de um único arquivo simples? Se os dados vierem de vários arquivos, você precisa de uma única etapa para todos os arquivos ou etapas separadas para cada arquivo para definir a lógica de pré-processamento? A decisão depende da complexidade das fontes de dados e de até que ponto elas são pré-processadas.

Em nossa implementação de referência, usamos o AWS Step Functions, que é um orquestrador de funções com tecnologia sem servidor, para definir as etapas do fluxo de trabalho. No entanto, a estrutura ML Max também oferece suporte a outros sistemas de pipeline ou máquina de estado, como o Apache AirFlow (consulte a seção Diferentes mecanismos para orquestração de pipelines) para impulsionar o desenvolvimento e a implantação de pipelines de ML.

Usar o SDK Step Functions

Para definir o pipeline de ML, primeiro usamos a API Python de alto nível fornecida pelo AWS Step Functions SDK Data Science (o SDK Step Functions) para definir dois componentes principais do pipeline: etapas e dados. Se você pensar em um pipeline como um gráfico acíclico direcionado (DAG), as etapas representam os nós no gráfico e os dados são mostrados como bordas

direcionadas que conectam um nó (uma etapa) ao próximo nó. Exemplos típicos de etapas de ML incluem pré-processamento, treinamento e avaliação. O Step Functions SDK fornece várias etapas integradas (como a TrainingStep) que você pode usar. Exemplos de dados incluem entrada, saída e muitos conjuntos de dados intermediários que são produzidos por algumas etapas do pipeline. Ao projetar um pipeline de ML, você não conhece os valores concretos dos itens de dados. Você pode definir espaços reservados para dados que sirvam como um modelo (semelhante aos parâmetros de função) e contenham somente o nome do item de dados e os tipos de dados primitivos. Dessa forma, você pode criar um esquema de pipeline completo sem conhecer com antecedência os valores concretos dos dados que se movem no gráfico. Para isso, você pode usar as classes de espaço reservado no SDK Step Functions para modelar explicitamente esses modelos de dados.

Os pipelines de ML também exigem parâmetros de configuração para realizar um controle detalhado do comportamento de cada etapa do ML. Esses espaços reservados de dados especiais são chamados de espaços reservados para parâmetros. Muitos de seus valores são desconhecidos quando você define o pipeline. Exemplos de espaços reservados para parâmetros incluem parâmetros relacionados à infraestrutura que você define durante o design do pipeline (por exemplo, Região da AWS ou URL da imagem do contêiner) e parâmetros relacionados à modelagem de ML (como hiperparâmetros) que você define ao executar o pipeline.

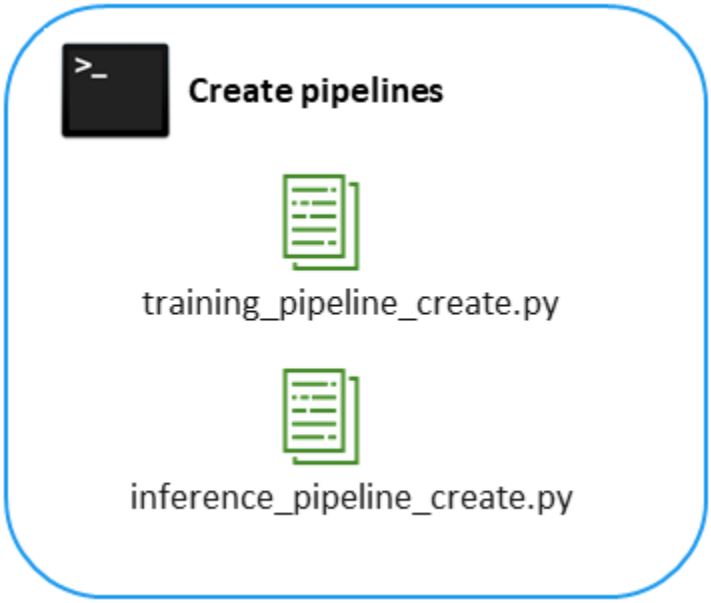
Estender o SDK Step Functions

Em nossa implementação de referência, um requisito era separar as definições do pipeline de ML da criação e implantação concretas do pipeline de ML usando configurações de parâmetros específicas. No entanto, algumas das etapas integradas no SDK Step Functions não permitiram que passássemos todos esses parâmetros de espaço reservado. Em vez disso, esperava-se que os valores dos parâmetros fossem obtidos diretamente durante o tempo de design do pipeline por meio de chamadas de API de configuração de SageMaker IA. Isso funciona bem se o ambiente de tempo de design da SageMaker IA for idêntico ao ambiente de tempo de execução da SageMaker IA, mas isso raramente é o caso em configurações do mundo real. Esse acoplamento forte entre o tempo de projeto do pipeline e o runtime e a suposição de que a infraestrutura da plataforma ML permanecerá constante dificultam significativamente a aplicabilidade do pipeline projetado. Na verdade, o pipeline de ML é interrompido imediatamente quando a plataforma de implantação subjacente sofre até mesmo a menor alteração.

Para superar esse desafio e produzir um pipeline robusto de ML (que queríamos projetar uma vez e executar em qualquer lugar), implementamos nossas próprias etapas personalizadas estendendo algumas das etapas integradas TrainingStep, incluindo ModelStep, e. TransformerStep Essas

extensões são fornecidas no projeto ML Max. As interfaces dessas etapas personalizadas oferecem suporte a muito mais espaços reservados de parâmetros que podem ser preenchidos com valores concretos durante a criação do pipeline ou quando o pipeline está em execução.

Etapa 4: Criar o pipeline



Depois de definir o pipeline de forma lógica, é hora de criar a infraestrutura para dar suporte ao pipeline. Essa etapa requer, no mínimo, os seguintes recursos:

- Armazenamento, para hospedar e gerenciar entradas e saídas do pipeline, incluindo código, artefatos de modelo e dados usados em treinamentos e execuções de inferência.
- Computação (GPU ou CPU) para modelagem e inferência, bem como pré-processamento e pós-processamento de dados.
- Orquestração, para gerenciar os recursos que estão sendo usados e programar qualquer execução regular. Por exemplo, o modelo pode ser retreinado periodicamente à medida que novos dados se tornam disponíveis.
- Execução de logs e alertas para monitorar a precisão do modelo de pipeline para a utilização de recursos e solução de problemas.

Implementação com AWS CloudFormation

Para criar o pipeline que usamos AWS CloudFormation, que é um AWS serviço para implantar e gerenciar a infraestrutura como código. Os AWS CloudFormation modelos incluem a definição do Step Functions que foi criada na etapa anterior com o SDK do Step Functions. Essa etapa inclui a criação da instância AWS-managed Step Functions, que é chamada de máquina de estado

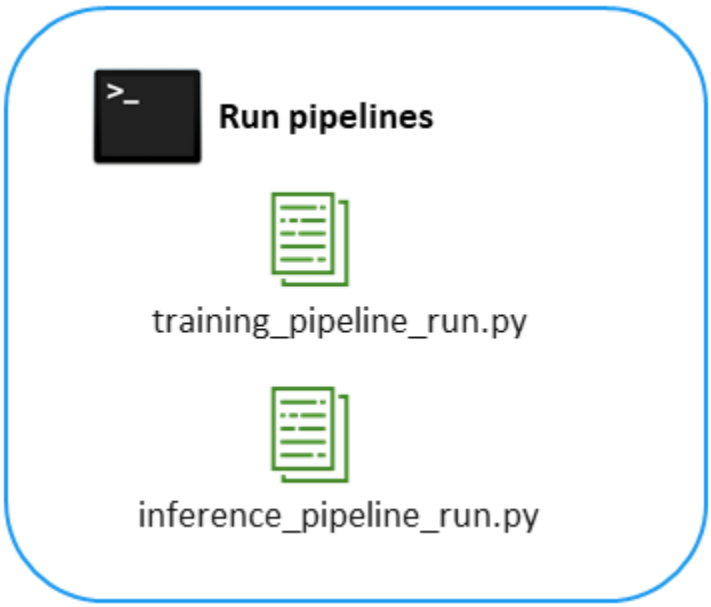
Step Functions. Nenhum recurso para treinamento e inferência é criado nesse estágio, porque os trabalhos de treinamento e inferência são executados sob demanda, somente quando são necessários, como trabalhos de SageMaker IA. Essa etapa também inclui a criação de funções AWS Identity and Access Management (IAM) para executar o Step Functions, executar a SageMaker IA e ler e gravar no Amazon S3.

Modificando a saída do SDK Funções de Etapa

Tivemos que fazer algumas pequenas modificações na CloudFormation saída da seção anterior. Usamos a correspondência simples de strings do Python para fazer o seguinte:

- Adicionamos lógica para criar a `Parameters` seção do CloudFormation modelo. Isso ocorre porque queremos criar duas funções e definir o nome do pipeline como um parâmetro junto com o ambiente de implantação. Essa etapa também abrange quaisquer recursos e funções adicionais que você queira criar, conforme discutido na etapa 6.
- Reformatamos três campos para ter o prefixo `!Sub` e as aspas necessárias para que possam ser atualizados dinamicamente como parte do processo de implantação:
  - A propriedade `StateMachineName`, que nomeia a máquina de estado.
  - A propriedade `DefinitionString`, que define a máquina de estado.
  - A propriedade `RoleArn`, que é retornada pela máquina de estado.

Etapa 5. Execute o pipeline



Essa etapa executa o pipeline de treinamento ou inferência que foi criado nas CloudFormation pilhas na etapa 4. O pipeline não pode ser executado até que seus parâmetros internos de espaço reservado tenham sido preenchidos com valores concretos. Essa ação de atribuir valores aos parâmetros de espaço reservado é a principal atividade da etapa 5. Exemplos de parâmetros de espaço reservado incluem:

- A localização dos conjuntos de dados de entrada, saída e intermediários
- A localização, no Amazon S3, dos scripts de runtime e outros códigos de pré-processamento ou avaliação que foram desenvolvidos na etapa 2 (por exemplo, `sm_submit_url` para o pipeline de treinamento)
- O nome da AWS região

Você precisa garantir que esses valores de caminho apontem para dados ou códigos válidos antes de executar o pipeline. Por exemplo, se você preencher o parâmetro de espaço reservado que representa o URL do Amazon S3 dos scripts de runtime do Python, você precisará fazer o upload desses scripts para esse URL. A pessoa que administra o pipeline é responsável pela verificação da consistência e pelo upload dos dados. As pessoas que definem ou criam o pipeline não precisam se preocupar com nada disso.

Dependendo da maturidade do pipeline, essa etapa pode ser automatizada para ser executada regularmente (semanal ou mensalmente). A automação também requer um monitoramento robusto, que é uma área importante, mas está fora do escopo deste guia. Para a execução do pipeline de treinamento, seria apropriado monitorar as métricas de avaliação. Para o pipeline de inferência, seria apropriado monitorar o desvio na distribuição dos dados de entrada e, se possível, coletar rótulos periodicamente e medir o desvio na precisão da previsão. Esses registros das execuções de treinamento e inferência devem ser registrados em um banco de dados para análise posterior.

Etapa 6. Expandir o pipeline

Este guia explica como você pode começar a criar pipelines de ML na AWS rapidamente, com uma arquitetura concreta. Há considerações adicionais para amadurecer o pipeline, como gerenciamento de metadados, rastreamento e monitoramento de experimentos. Esses são tópicos importantes que estão fora do escopo deste guia. As seções a seguir discutem outro aspecto do gerenciamento de pipeline, que é a automação do pipeline.

Diferentes níveis de automação

Embora você possa configurar um pipeline de treinamento manualmente no console do SageMaker AI, na prática, recomendamos minimizar os pontos de contato manuais na implantação dos canais de treinamento de ML para garantir que os modelos de ML sejam implantados de forma consistente e repetida. Dependendo dos seus requisitos e dos problemas empresariais que você está abordando, você pode determinar e usar uma estratégia de implantação em três níveis: semiautomatizado, totalmente automatizado e totalmente gerenciado.

- Semiautomatizado — Por padrão, as etapas discutidas na seção anterior seguem uma abordagem semiautomatizada, pois implantam o pipeline de treinamento e inferência usando modelos CloudFormation. Isso ajuda a garantir a reprodutibilidade do pipeline e contribui para que você possa alterá-lo e atualizá-lo facilmente.
- Totalmente automatizado — Uma opção mais avançada é usar a integração contínua e a implantação contínua (CI/CD) nos ambientes de desenvolvimento, preparação e produção. A incorporação de práticas de CI/CD à implantação do pipeline de treinamento pode garantir que a automação inclua rastreabilidade e portas de qualidade.
- Totalmente gerenciado — Em última análise, você pode desenvolver um sistema totalmente gerenciado para implantar um pipeline de treinamento de ML com um conjunto de manifestos simples, e o sistema pode se autoconfigurar e coordenar os serviços AWS necessários.

Neste guia, optamos por apresentar uma arquitetura concreta. No entanto, existem tecnologias alternativas que você pode considerar. As próximas duas seções discutem algumas escolhas alternativas para a plataforma e o mecanismo de orquestração.

Diferentes plataformas para workloads de ML

O Amazon SageMaker AI é o serviço gerenciado da AWS para treinar e distribuir modelos de ML. Muitos usuários apreciam sua ampla variedade de recursos integrados e as muitas opções que ele oferece para executar workloads de ML. O SageMaker AI é particularmente útil se você está começando a implementar ML na nuvem. Os principais recursos do SageMaker AI incluem:

- Rastreabilidade integrada (incluindo rotulagem, treinamento, rastreamento de modelos, otimização e inferência).
- Opções integradas com apenas um clique para treinamento e inferência com um mínimo de experiência em Python e ML.
- Ajuste avançado de hiperparâmetros.
- Suporte a todos as estruturas importantes de inteligência artificial e machine learning (ML/AI) e contêineres Docker personalizados.
- Recursos de monitoramento integrados.
- Rastreamento integrado de históricos, incluindo trabalhos de treinamento, trabalhos de processamento, trabalhos de transformação em lote, modelos, endpoints e capacidade de pesquisa. Alguns históricos, como treinamento, processamento e transformação em lote, são imutáveis e só podem ser anexados.

Uma das alternativas ao uso do SageMaker AI é o AWS Batch. O AWS Batch proporciona um nível mais baixo de controle sobre a computação e a orquestração do seu ambiente, mas não é personalizado para machine learning. Alguns dos seus principais recursos incluem:

- Ajuste de escala automática pronta para uso dos recursos de computação com base na workload.
- Suporte pronto para uso para prioridade da tarefa, novas tentativas e dependências do trabalho.
- Abordagem baseada em filas que oferece suporte à criação de trabalhos recorrentes e sob demanda.
- Suporte para workloads da CPU e GPU. A capacidade de usar a GPU para compilar modelos de ML é essencial, pois a GPU pode acelerar significativamente o processo de treinamento, especialmente para modelos de aprendizado profundo.
- Capacidade de definir uma imagem de máquina da Amazon (AMI) personalizada para o ambiente computacional.

Diferentes mecanismos para orquestração do pipeline

O segundo componente principal é a camada de orquestração do pipeline. A AWS fornece Step Functions para uma experiência de orquestração totalmente gerenciada. Uma alternativa popular ao Step Functions é o Apache Airflow. Ao tomar uma decisão entre os dois, considere o seguinte:

- Infraestrutura necessária — O AWS Step Functions é um serviço totalmente gerenciado e com tecnologia sem servidor, enquanto o Airflow exige o gerenciamento de sua própria infraestrutura e é baseado em software de código aberto. Em consequência, o Step Functions fornece alta disponibilidade pronta para uso, enquanto a administração do Apache Airflow exige etapas adicionais.
- Recursos de agendamento — Tanto o Step Functions quanto o Airflow oferecem funcionalidades comparáveis.
- Recursos de visualização e interface de usuário — Tanto o Step Functions quanto o Airflow oferecem funcionalidades comparáveis.
- Passando variáveis dentro do gráfico computacional — O Step Functions fornece funcionalidade limitada para o uso de funções AWS Lambda, enquanto o Airflow fornece interfaces xCOM.
- Uso — O Step Functions é muito popular entre clientes AWS, e o Airflow tem sido amplamente adotado pela comunidade de engenharia de dados.

Conclusão

À medida que o machine learning passa de uma disciplina de pesquisa para um campo aplicado, observamos um crescimento anual de 25% no desenvolvimento, implantação e operação do pipeline de ML em vários setores. O valor comercial do ML é obtido por meio de operações e pipelines diários de ML, que, por sua vez, impulsionam a pesquisa e o desenvolvimento de modelos e algoritmos de ML. No entanto, a implantação do ML na produção apresenta vários desafios, pois entrelaça atividades e artefatos significativamente diferentes, como gerenciamento, processamento, análise, modelagem, verificação e segurança de dados. Por meio de vários contratos de IA/ML com clientes AWS, nossa equipe de ciência de dados observou que um dos principais desafios é a falta de um fluxo de trabalho de ponta a ponta que forneça um conjunto de modelos para fundir ou separar de forma ideal diferentes atividades e artefatos do ML DevOps. Neste guia, apresentamos o fluxo de trabalho do ML Max para resolver esse problema urgente. O ML Max fornece diretrizes detalhadas e um conjunto de modelos de programação. O objetivo é permitir uma transição rápida e econômica de uma fase de desenvolvimento de modelo interativo para uma configuração de pipeline de ML completa e escalonável que esteja pronta para produção.

Recursos adicionais

Guias e padrões relacionados

- Quantificar a incerteza em sistemas de aprendizado profundo
- Migre o ML Build, treine e implante workloads para o Amazon SageMaker AI usando ferramentas para desenvolvedores AWS
- Site de recomendações da AWS

AWS serviços da

- AWS Batch
- AWS CloudFormation
- IAM no
- Amazon SageMaker AI
- AWS Step Functions

Recursos gerais da AWS

- Documentação da AWS
- AWS Referência geral do
- AWS Glossário da

Histórico do documento

A tabela a seguir descreve alterações significativas feitas neste guia. Se desejar receber notificações sobre futuras atualizações, inscreva-se em um feed RSS.

Alteração	Descrição	Data
Publicação inicial	—	29 de janeiro de 2021

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.