Creación de canalizaciones de aprendizaje automático listas para la producción en AWS

AWS Orientación prescriptiva

AWS Orientación prescriptiva: Creación de canalizaciones de aprendizaje automático listas para la producción en AWS

Table of Contents

Creación de canalizaciones de aprendizaje automático listas para la producción en AWS

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

Enero de 2021 (historial de documentos)

Los proyectos de machine learning (ML) requieren un esfuerzo significativo de varias etapas que incluye el modelado, la implementación y la producción para ofrecer valor empresarial y resolver problemas del mundo real. Hay numerosas alternativas y opciones de personalización disponibles en cada paso, lo que hace que sea cada vez más difícil preparar un modelo de ML para la producción dentro de las limitaciones de recursos y presupuesto. Durante los últimos años en Amazon Web Services (AWS), nuestro equipo de ciencia de datos ha trabajado con diferentes sectores de la industria en iniciativas de ML. Identificamos los problemas que muchos AWS clientes comparten y que se deben tanto a problemas organizativos como a desafíos técnicos, y hemos desarrollado un enfoque óptimo para ofrecer soluciones de aprendizaje automático listas para la producción.

Esta guía es para científicos de datos e ingenieros de ML que participan en las implementaciones en implementaciones de canalización de ML. Describe nuestro enfoque para ofrecer canalizaciones de ML listas para la producción. La guía explica cómo puede pasar de ejecutar modelos de ML de forma interactiva (durante el desarrollo) a implementarlos como parte de una canalización (durante la producción) para su caso de uso de ML. Para ello, también hemos desarrollado un conjunto de plantillas de ejemplo (consulte el proyecto ML Max), a fin de acelerar la entrega de soluciones de ML personalizadas a la producción, de modo que pueda empezar rápidamente sin tener que elegir demasiadas opciones de diseño.

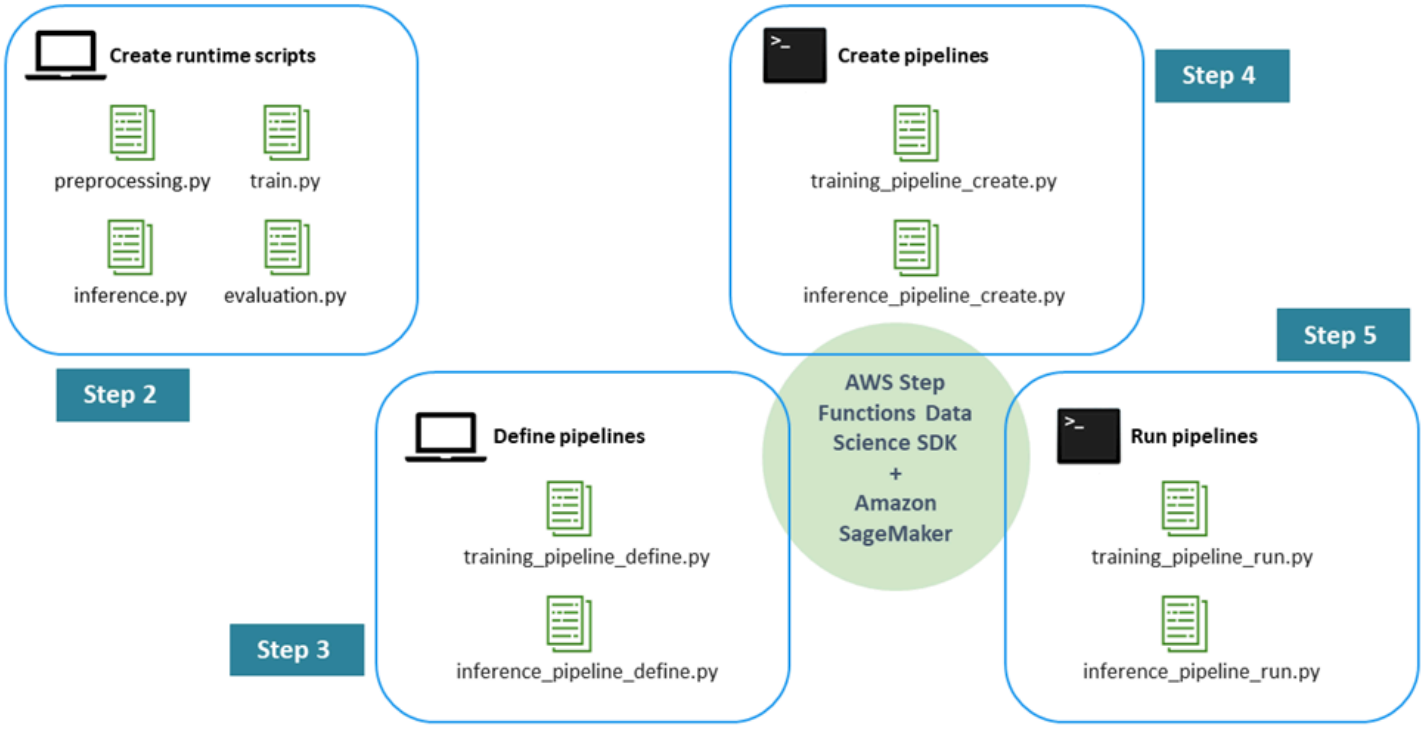
Descripción general de

El proceso para crear una canalización de ML lista para la producción consta de los siguientes pasos:

- Paso 1. Realice el EDA y desarrolle el modelo inicial: los científicos de datos hacen que los datos sin procesar estén disponibles en Amazon Simple Storage Service (Amazon S3), realizan análisis de datos exploratorios (EDA), desarrollan el modelo de aprendizaje automático inicial y evalúan su rendimiento de inferencia. Puede realizar estas actividades de forma interactiva a través de los cuadernos de Jupyter.

- Paso 2. Cree los scripts en tiempo de ejecución: integre el modelo con los scripts de Python en tiempo de ejecución para que pueda administrarse y aprovisionarse mediante un marco de aprendizaje automático (en nuestro caso, Amazon SageMaker AI). Este es el primer paso para pasar del desarrollo interactivo de un modelo independiente al de producción. En concreto, usted define la lógica para el preprocesamiento, la evaluación, el entrenamiento y la inferencia por separado.
- Paso 3. Defina la canalización: defina los marcadores de entrada y salida para cada paso de la canalización. Los valores concretos para estos valores se proporcionarán más adelante, durante el tiempo de ejecución (paso 5). Usted se centra en los procesos de entrenamiento, inferencia, validación cruzada y pruebas retrospectivas.
- Paso 4. Cree la canalización: cree la infraestructura subyacente, incluida la instancia de la máquina de AWS Step Functions estado, de forma automatizada (casi con un clic), mediante el uso de. AWS CloudFormation
- Paso 5. Ejecute la canalización: usted ejecuta la canalización definida en el paso 4. También prepara los metadatos y los datos o las ubicaciones de los datos para rellenar valores concretos para los input/output marcadores de posición que definió en el paso 3. Esto incluye los scripts del tiempo de ejecución definidos en el paso 2, así como los hiperparámetros del modelo.
- Paso 6. Amplíe la cartera: implemente procesos de integración y despliegue continuos (CI/CD), readiestramiento automatizado, inferencia programada y ampliaciones similares de la canalización.

El siguiente diagrama ilustra los principales pasos de este proceso.



Steps for creating the training and inference pipeline

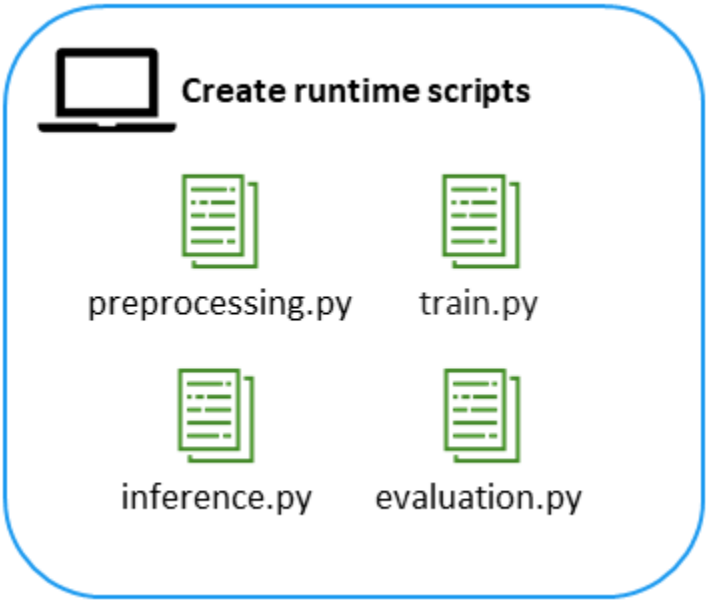
Paso 1. Realice la EDA y desarrolle el modelo inicial

En este paso, los científicos de datos realizan un análisis de datos exploratorio (EDA) para comprender el caso de uso y los datos del ML. A continuación, desarrollan los modelos de ML (por ejemplo, modelos de clasificación y regresión) para resolver el problema en un caso de uso determinado. Durante el desarrollo del modelo, el científico de datos suele hacer suposiciones sobre las entradas y las salidas, como los formatos de los datos, el ciclo de vida de los datos y las ubicaciones de los resultados intermedios. Estas suposiciones deben documentarse para que puedan usarse para la verificación durante las pruebas unitarias del paso 2.

Si bien este paso se centra en el desarrollo del modelo, los científicos de datos suelen tener que escribir una cantidad mínima de código auxiliar para el preprocesamiento, la formación, la evaluación y la inferencia. El científico de datos debería poder ejecutar este código en el entorno de desarrollo. También recomendamos facilitar argumentos de tiempo de ejecución opcionales para que este código auxiliar se pueda configurar dinámicamente para que se ejecute en otros entornos sin necesidad de realizar cambios manuales exhaustivos. Esto acelerará la integración entre el modelo y la canalización en los pasos 2 y 3. Por ejemplo, el código para leer los datos sin procesar debe encapsularse en funciones para que los datos se puedan preprocesar de manera coherente.

Le recomendamos que comience con un marco como scikit-learn, XGBoost, PyTorch, Keras o TensorFlow para desarrollar el modelo ML y su código auxiliar. Por ejemplo, scikit-learn es una biblioteca de ML gratuita escrita en Python. Facilita una convención de API uniforme para los objetos e incluye cuatro objetos principales (estimador, predictor, transformador y modelo) que abordan transformaciones de datos ligeras, admiten la ingeniería de características y etiquetas, y encapsulan los pasos de preprocesamiento y modelado. Estos objetos ayudan a evitar la proliferación de códigos reutilizables y evitan que los datos de validación y prueba se filtren al conjunto de datos de entrenamiento. Del mismo modo, cada marco de ML tiene su propia implementación de los elementos clave del ML, y le recomendamos que cumpla con las convenciones de API del marco que haya seleccionado al desarrollar modelos de ML.

Paso 2. Cree los scripts de tiempo de ejecución



En este paso, usted integrará el modelo que desarrolló en el paso 1 y su código auxiliar asociado en una plataforma de ML para el entrenamiento y la inferencia listas para la producción. Específicamente, esto implica el desarrollo de scripts de tiempo de ejecución para que el modelo pueda incorporarse a la SageMaker IA. Estos scripts de Python independientes incluyen funciones de devolución de llamada de SageMaker IA predefinidas y variables de entorno. Se ejecutan dentro de un contenedor de SageMaker IA alojado en una instancia de Amazon Elastic Compute Cloud (Amazon EC2). La documentación del SDK para Python de Amazon SageMaker AI proporciona información detallada sobre cómo estas devoluciones de llamada y la configuración auxiliar funcionan juntas para el entrenamiento y la inferencia. En las siguientes secciones se proporcionan recomendaciones adicionales para desarrollar scripts de tiempo de ejecución de ML, en función de nuestra experiencia trabajando con clientes. AWS

Usando los trabajos de procesamiento

SageMaker La IA ofrece dos opciones para realizar inferencias de modelos en modo lote. Puede utilizar un trabajo de procesamiento de SageMaker IA o un trabajo de transformación por lotes. Cada opción tiene ventajas y desventajas.

Un trabajo de procesamiento consiste en un archivo de Python que se ejecuta dentro de un contenedor de SageMaker IA. El trabajo de procesamiento consiste en cualquier lógica que ponga en su archivo de Python. Ofrece estas ventajas:

- Cuando se comprende la lógica básica de un trabajo de entrenamiento, los trabajos de procesamiento son fáciles de configurar y entender. Comparten las mismas abstracciones que los trabajos de entrenamiento (por ejemplo, ajustar el número de instancias y la distribución de datos).
- Los científicos de datos y los ingenieros de ML tienen el control total sobre las opciones de manipulación de datos.
- El científico de datos no tiene que gestionar la lógica de ningún I/O componente, excepto una read/write funcionalidad familiar.
- Es un poco más fácil ejecutar los archivos en entornos que no sean de SageMaker IA, lo que facilita el desarrollo rápido y las pruebas locales.
- Si se produce un error, un trabajo de procesamiento falla en cuanto el script falla y no hay que esperar inesperadamente para volver a intentarlo.

Por otro lado, los trabajos de transformación por lotes son una extensión del concepto de punto final de SageMaker IA. En tiempo de ejecución, estos trabajos importan funciones de devolución de llamada, que luego se encargan de leer los datos, cargar el modelo y hacer las predicciones. I/O Los trabajos de transformación por lotes tienen las siguientes ventajas:

- Utilizan una abstracción de distribución de datos que difiere de la abstracción utilizada en los trabajos de formación.
- Utilizan la misma estructura básica de archivos y funciones tanto para la inferencia por lotes como para la inferencia en tiempo real, lo cual resulta práctico.
- Tienen un mecanismo integrado de tolerancia a errores basado en intentos. Por ejemplo, si se produce un error en un lote de registros, se volverá a intentar varias veces antes de que el trabajo finalice por error.

Debido a su transparencia, su facilidad de uso en varios entornos y su abstracción compartida con los trabajos de entrenamiento, decidimos utilizar el trabajo de procesamiento en lugar del trabajo de transformación por lotes en la arquitectura de referencia que se presenta en estas recomendaciones.

Debe ejecutar los scripts de tiempo de ejecución de Python localmente antes de implementarlos en la nube. En concreto, le recomendamos que utilice la cláusula de protección principal al estructurar sus scripts de Python y realizar pruebas unitarias.

Uso de la cláusula de protección principal

Utilice una cláusula de protección principal para admitir la importación de módulos y ejecutar su script de Python. Ejecutar scripts de Python de forma individual es beneficioso para depurar y aislar problemas en la canalización de ML. Recomendamos los siguientes pasos:

- Utilice un analizador de argumentos en los archivos de procesamiento de Python para especificar input/output los archivos y sus ubicaciones.
- Proporcione una guía principal y funciones de prueba para cada archivo de Python.
- Después de probar un archivo de Python, incorpóralo en las diferentes etapas del proceso de aprendizaje automático, ya sea que utilices un AWS Step Functions modelo o un trabajo de procesamiento de SageMaker IA.
- Utilice las declaraciones Assert en las secciones críticas del script para facilitar las pruebas y la depuración. Por ejemplo, puede usar una declaración Assert para garantizar que el número de características del conjunto de datos sea uniforme después de la carga.

Pruebas unitarias

Las pruebas unitarias de los scripts en tiempo de ejecución que se escribieron para la canalización son una tarea importante que, con frecuencia, se pasa por alto en el desarrollo de la canalización de ML. Esto se debe a que el machine learning y la ciencia de datos son campos relativamente nuevos y han tardado en adoptar prácticas de ingeniería de software bien establecidas, como las pruebas unitarias. Como la canalización de ML se utilizará en el entorno de producción, es esencial probar el código de la canalización antes de aplicar el modelo de ML a aplicaciones del mundo real.

Las pruebas unitarias del script en tiempo de ejecución también proporcionan las siguientes ventajas exclusivas para los modelos de ML:

- Evita las transformaciones de datos inesperadas. La mayoría de las canalizaciones de ML implican muchas transformaciones de datos, por lo que es fundamental que estas transformaciones se realicen según lo esperado.
- Valida la reproducibilidad del código. Cualquier aleatoriedad en el código se puede detectar mediante pruebas unitarias con diferentes casos de uso.
- Refuerza la modularidad del código. Las pruebas unitarias suelen estar asociadas a la medida de cobertura de la prueba, que es el grado en que un conjunto de pruebas en particular (una colección de casos de prueba) ejecuta el código fuente de un programa. Para lograr una alta

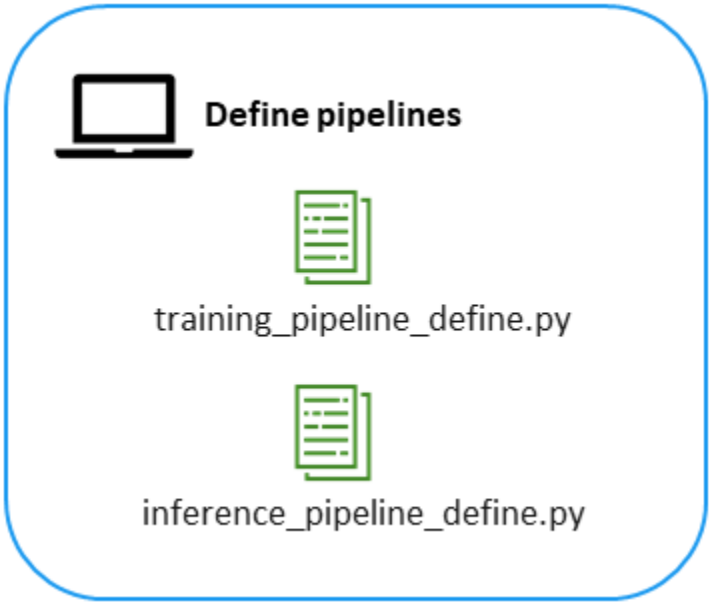
cobertura de pruebas, los desarrolladores modularizan el código, ya que es difícil escribir pruebas unitarias para una gran cantidad de código sin dividirlo en funciones o clases.

- Evita que se introduzcan errores o código de baja calidad en la producción.

Le recomendamos que utilice un marco de pruebas unitarias maduro, como pytest, para escribir los casos de las pruebas unitarias, ya que es más fácil gestionar pruebas unitarias exhaustivas dentro de un marco.

> **Important**
>
> Las pruebas unitarias no pueden garantizar que se prueben todos los casos extremos, pero pueden ayudarle a evitar errores de forma proactiva antes de implementar el modelo. Le recomendamos que también monitorice el modelo después de la implementación para garantizar la excelencia operativa.

Paso 3. Definir la canalización



En este paso se definen la secuencia y la lógica de las acciones que se llevarán a cabo en el proceso. Esto incluye pasos discretos, así como sus entradas y salidas lógicas. Por ejemplo, ¿cuál es el estado de los datos al principio del proceso? ¿Provienen de varios archivos con distintos niveles de detalle, o bien de un único archivo plano? Si los datos provienen de varios archivos, ¿necesita un solo paso para todos los archivos, o bien pasos separados para cada archivo a fin de definir la lógica de preprocesamiento? La decisión dependerá de la complejidad de los orígenes de datos y del grado en que estén preprocesados.

En nuestra implementación de referencia usamos AWS Step Functions, un orquestador de funciones sin servidor, para definir los pasos del flujo de trabajo. Sin embargo, el marco ML Max también es compatible con otros sistemas de canalización o máquinas de estado, como Apache AirFlow (consulte la sección Diferentes motores para la orquestación de canalizaciones) para impulsar el desarrollo y la implementación de canalizaciones de ML.

Uso del SDK de Step Functions

Para definir el proceso de ML, usaremos en primer lugar la API de alto nivel de Python, proporcionada por el SDK de AWS Step Functions Data Science (el SDK de Step Functions), para definir dos componentes clave del proceso: los pasos y los datos. Si consideramos el proceso como un gráfico acíclico dirigido (DAG), los pasos representan los nodos del gráfico y los datos se muestran como periferias dirigidas que conectan un nodo (paso) con el siguiente. Algunos ejemplos

típicos de pasos de ML son el preprocesamiento, la formación y la evaluación. El SDK de Step Functions proporciona una serie de pasos integrados (como el TrainingStep) que puede utilizar. Como ejemplos de datos tenemos la entrada, la salida y muchos conjuntos de datos intermedios que se generan a partir de algunos pasos del proceso. Cuando diseña un proceso de ML, no conoce los valores concretos de los elementos de datos. Puede definir marcadores de posición de datos que sirvan como plantilla (similares a los parámetros de una función) y que contengan solo el nombre del elemento de datos y los tipos de datos primitivos. Este método le permite diseñar un esquema de proceso completo sin conocer de antemano los valores concretos de los datos que circulan por el gráfico. Puede usar las clases de marcadores de posición del SDK de Step Functions para modelar de forma explícita estas plantillas de datos.

Los procesos de ML también requieren parámetros de configuración para controlar, de forma pormenorizada, el comportamiento de cada paso de ML. Estos marcadores de posición de datos especiales se denominan marcadores de posición de parámetros. Muchos de sus valores se desconocen al definir el proceso. Entre los ejemplos de marcadores de posición de parámetros se incluyen los parámetros relacionados con la infraestructura que se definen durante el diseño de la canalización (por ejemplo, Región de AWS la URL de la imagen del contenedor) y los parámetros relacionados con el modelado de aprendizaje automático (como los hiperparámetros) que se definen al ejecutar la canalización.

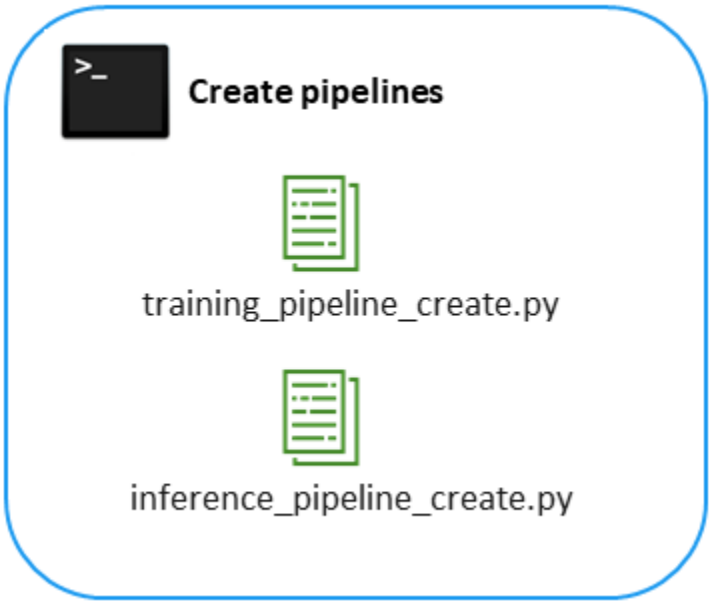
Ampliación del SDK de Step Functions

En nuestra implementación de referencia, uno de los requisitos era separar las definiciones de procesos de ML de la creación e implementación de procesos de ML concretos mediante ajustes de parámetros específicos. Sin embargo, algunos de los pasos integrados en el SDK de Step Functions no nos permitían introducir todos estos parámetros de marcador de posición. En cambio, se esperaba que los valores de los parámetros se obtuvieran directamente durante el diseño de la canalización mediante llamadas a la API de configuración de la IA. SageMaker Esto funciona bien si el entorno de tiempo de diseño de la SageMaker IA es idéntico al entorno de ejecución de la SageMaker IA, pero esto rara vez ocurre en entornos del mundo real. El acoplamiento ajustado entre el tiempo de diseño y el tiempo de ejecución del proceso, así como la suposición de que la infraestructura de la plataforma de ML se mantendrá constante, dificultan considerablemente la aplicabilidad del proceso diseñado. De hecho, el proceso de ML se interrumpe de inmediato cuando la plataforma de implementación subyacente sufre el más mínimo cambio.

Para superar este desafío y crear un proceso de aprendizaje automático sólido (que queríamos diseñar una vez y ejecutar en cualquier lugar), implementamos nuestros propios pasos

personalizados ampliando algunos de los pasos integrados, incluidos TrainingStepModelStep, y. TransformerStep Estas extensiones se incluyen en el proyecto ML Max. Las interfaces de estos pasos personalizados admiten muchos más marcadores de parámetros, que se pueden rellenar con valores concretos durante la creación del proceso o bien durante su ejecución.

Paso 4. Crear la canalización



Tras definir la canalización de forma lógica, es hora de crear la infraestructura que la respalde. Este paso requiere, como mínimo, las siguientes capacidades:

- Almacenamiento, para alojar y gestionar las entradas y salidas de la canalización, incluidos el código, los artefactos del modelo y los datos utilizados en las ejecuciones de entrenamiento e inferencia.
- Computación (GPU o CPU), para el modelado y la inferencia, así como para el preprocesamiento y posprocesamiento de datos.
- Orquestación para gestionar los recursos que se utilizan y programar cualquier ejecución regular. Por ejemplo, el modelo podría reentrenarse periódicamente a medida que se disponga de nuevos datos.
- Registro y alertas para supervisar la precisión del modelo de canalización, el uso de los recursos y la solución de problemas.

Implementación con AWS CloudFormation

Para crear la canalización que utilizamos AWS CloudFormation, que es un AWS servicio para implementar y administrar la infraestructura como código. Las AWS CloudFormation plantillas incluyen la definición de Step Functions que se creó en el paso anterior con el SDK de Step

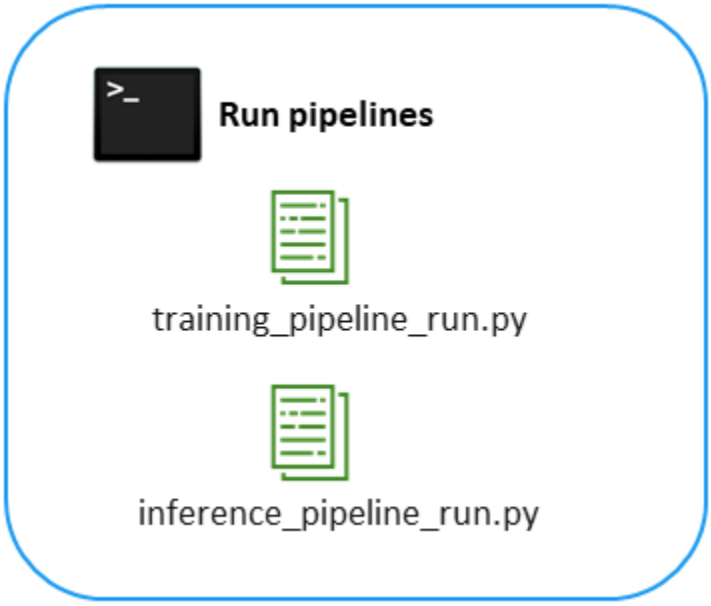
Functions. Este paso incluye la creación de la instancia de AWS-managed Step Functions, que se denomina máquina de estados de Step Functions. En este momento no se crean recursos para la formación y la inferencia, ya que los trabajos de formación e inferencia se ejecutan a pedido, solo cuando son necesarios, como trabajos de SageMaker IA. Este paso también incluye la creación de funciones AWS Identity and Access Management (IAM) para ejecutar Step Functions, ejecutar SageMaker IA y leer y escribir desde Amazon S3.

Modificación de la salida del SDK de Step Functions

Tuvimos que hacer algunas modificaciones menores en el CloudFormation resultado de la sección anterior. Usamos una simple coincidencia de cadenas de Python para hacer lo siguiente:

- Hemos añadido la lógica para crear la `Parameters` sección de la CloudFormation plantilla. Esto se debe a que queremos crear dos funciones y definir el nombre de la canalización como parámetro junto con el entorno de implementación. En este paso también se incluyen los recursos y funciones adicionales que usted podría querer crear, tal y como se explica en el paso 6.
- Reformateamos tres campos para que tuvieran el `!Sub` prefijo y las comillas necesarios para que se puedan actualizar de forma dinámica como parte del proceso de implementación:
  - La `StateMachineName` propiedad, que nombra a la máquina de estados.
  - La `DefinitionString` propiedad, que define la máquina de estados.
  - La `RoleArn` propiedad, que devuelve la máquina de estados.

Paso 5. Ejecute la canalización



Este paso ejecuta la canalización de entrenamiento o inferencia que se creó en las CloudFormation pilas en el paso 4. La canalización no se puede ejecutar hasta que sus parámetros de marcador de posición internos se hayan rellenado con valores concretos. Esta acción de asignar valores a los parámetros de los marcadores de posición es la actividad principal del paso 5. Algunos ejemplos de parámetros de marcador de posición son:

- La ubicación de los conjuntos de datos de entrada, salida e intermedios
- La ubicación en Amazon S3 de los scripts de tiempo de ejecución y otro código de preprocesamiento o evaluación que se desarrolló en el paso 2 (por ejemplo, `sm_submit_url` para el proceso de formación)
- El nombre de la región AWS

Debe asegurarse de que estos valores de trayectoria apuntan a datos o códigos válidos antes de ejecutar la canalización. Por ejemplo, si rellena el parámetro de marcador de posición que representa la URL de Amazon S3 de los scripts en tiempo de ejecución de Python, debe cargar esos scripts en esa URL. La persona que dirige la canalización es responsable de comprobar la coherencia y cargar los datos. Las personas que definen o crean la canalización no tienen que preocuparse por nada de esto.

En función de la madurez de la canalización, este paso puede automatizarse para que se ejecute de forma regular (semanal o mensual). La automatización también requiere una monitorización sólida, que es un área importante, pero queda fuera del alcance de estas recomendaciones. Para el proceso de entrenamiento, sería apropiado monitorizar las métricas de evaluación. Para la canalización de inferencia, sería apropiado monitorizar la desviación en la distribución de los datos de entrada y, si es posible, recopilar etiquetas periódicamente y medir la desviación en la precisión de la predicción. Estos registros de las series de entrenamiento e inferencia deben registrarse en una base de datos para analizarlos más adelante.

Paso 6. Ampliar la canalización

En esta guía, se explica cómo puede empezar a crear canalizaciones de ML en AWS de forma rápida y con una arquitectura concreta. Existen consideraciones adicionales para mejorar la canalización, como la gestión de metadatos, el seguimiento de los experimentos y la monitorización. Estos son temas importantes que quedan fuera del alcance de estas recomendaciones. En las siguientes secciones se analiza otro aspecto de la gestión de canalizaciones, que es la automatización de las canalizaciones.

Diferentes niveles de automatización

Aunque puede configurar un proceso de entrenamiento manualmente en la consola de SageMaker AI, en la práctica, le recomendamos minimizar los puntos de contacto manuales en la implementación de los procesos de ML para garantizar que los modelos de ML se implementen de forma coherente y repetida. En función de sus requisitos y de los problemas empresariales a los que se dirija, puede determinar e implementar una estrategia de implementación en tres niveles: semiautomática, totalmente automatizada y totalmente gestionada.

- Semiautomática: de forma predeterminada, los pasos descritos en la sección anterior siguen un enfoque semiautomatizado, ya que implementan el proceso de entrenamiento e inferencia mediante plantillas de CloudFormation. Esto ayuda a garantizar la reproducibilidad de la canalización y permite cambiarla y actualizarla fácilmente.
- Totalmente automatizada: una opción más avanzada es utilizar la integración y la implementación continuas (CI/CD) en los entornos de desarrollo, puesta en escena y producción. La incorporación de las prácticas de CI/CD a la implementación de la canalización de entrenamiento puede garantizar que la automatización incluya la trazabilidad y las puertas de calidad.
- Totalmente gestionado: en última instancia, puede desarrollar un sistema totalmente gestionado para poder implementar un proceso de entrenamiento en ML con un conjunto de sencillos manifiestos, y el sistema podrá autoconfigurar y coordinar los servicios necesarios de AWS.

En estas recomendaciones elegimos presentar una arquitectura concreta. Sin embargo, hay tecnologías alternativas que puede considerar. En las dos secciones siguientes se analizan algunas opciones alternativas para la plataforma y el motor de orquestación.

Diferentes plataformas para cargas de trabajo de ML

Amazon SageMaker AI es el servicio de AWS gestionado para el entrenamiento y el suministro de modelos de ML. Muchos usuarios aprecian su amplia gama de características integradas y las numerosas opciones que ofrece para ejecutar cargas de trabajo de ML. SageMaker AIresulta especialmente útil si acaba de empezar a implementar el ML en la nube. Las principales características de SageMaker AI incluyen:

- Trazabilidad integrada (que incluye el etiquetado, el entrenamiento, el seguimiento de modelos, la optimización y la inferencia).
- Opciones integradas de un solo clic para entrenamiento e inferencia con una experiencia mínima en Python y ML.
- Ajuste avanzado de hiperparámetros.
- Soporte para los principales marcos de inteligencia artificial y machine learning (ML/AI) y contenedores Docker personalizados.
- Capacidades de monitorización integradas.
- Seguimiento integrado de los historiales, incluidos los trabajos de entrenamiento, los trabajos de procesamiento, los trabajos de transformación por lotes, los modelos, los puntos de conexión y la capacidad de búsqueda. Algunos historiales, como los de entrenamiento, procesamiento y transformación por lotes, son inmutables y solo se pueden adjuntar.

Una de las alternativas al uso de SageMaker AI es AWS Batch. AWS Batch proporciona un nivel inferior de control sobre el procesamiento y la orquestación de su entorno, pero no está diseñado a medida para el machine learning. Algunas de sus características principales incluyen:

- Escalado automático diferente de los recursos informáticos para usar en función de la carga de trabajo.
- Soporte diferente para la prioridad de las tareas, los reintentos y las dependencias entre las tareas.
- Enfoque basado en colas que permite crear trabajos recurrentes y según demanda.
- Soporte para cargas de trabajo de CPU y GPU. La capacidad de usar la GPU para crear modelos de ML es fundamental, ya que la GPU puede acelerar el proceso de entrenamiento de manera significativa, especialmente en el caso de los modelos de aprendizaje profundo.
- Capacidad de definir una imagen de máquina de Amazon (AMI) personalizada para el entorno informático.

Diferentes motores para la orquestación de canalizaciones

El segundo componente principal es la capa de orquestación de canalizaciones. AWSproporciona Step Functions (funciones de paso) para una experiencia de orquestación totalmente gestionada. Una alternativa popular a Step Functions es Apache Airflow. Cuando tome una decisión entre los dos, tenga en cuenta los siguientes factores:

- Infraestructura requerida: AWS Step Functions es un servicio totalmente gestionado y no tiene servidores, mientras que Airflow requiere la administración de su propia infraestructura y se basa en un software de código abierto. Como resultado, Step Functions proporciona una alta disponibilidad desde el primer momento, mientras que la administración de Apache Airflow requiere pasos adicionales.
- Capacidades de programación: tanto Step Functions como Airflow ofrecen funcionalidades comparables.
- Capacidades de visualización e interfaz de usuario: tanto Step Functions como Airflow ofrecen funcionalidades comparables.
- Pasar variables dentro del gráfico computacional: Step Functions proporciona una funcionalidad limitada para el uso de AWS Lambda funciones, mientras que Airflow proporciona interfaces XCom.
- Uso: Step Functions es muy popular entre AWS los clientes y Airflow ha sido ampliamente adoptado por la comunidad de ingenieros de datos.

Conclusión

A medida que el machine learning pase de ser una disciplina de investigación a un campo aplicado, hemos observado un crecimiento anual del 25 por ciento en el desarrollo, la implementación y la operación de los proyectos de ML en varios sectores. El valor empresarial del ML se materializa a través de las operaciones y los procesos diarios de ML, que, a su vez, impulsan la investigación y el desarrollo de modelos y algoritmos de ML. Sin embargo, la implementación del ML en la producción presenta numerosos desafíos, ya que entrelaza actividades y artefactos muy diferentes, como la administración, el procesamiento, el análisis, el modelado, la verificación y la seguridad de los datos. A través de numerosos compromisos de IA/ML con los clientes de AWS, nuestro equipo de ciencia de datos ha observado que un desafío clave es la falta de un flujo de trabajo integral que proporcione un conjunto de plantillas para fusionar o separar de manera óptima las diferentes actividades y artefactos de DevOps de ML. En esta guía, presentamos el flujo de trabajo de ML Max para abordar este problema urgente. ML Max ofrece instrucciones paso a paso y un conjunto de plantillas de programación. El objetivo es permitir una transición rápida y rentable de una fase de desarrollo de modelos interactivos a una configuración de canalización de ML completa y escalable que esté lista para la producción.

Recursos adicionales

Guías y patrones relacionados

- Cuantificación de la incertidumbre en los sistemas de aprendizaje profundo
- Migre cargas de trabajo de machine learning (ML) para crear, entrenar e implementar cargas de trabajo a Amazon SageMaker AI mediante herramientas para desarrolladores de AWS
- Sitio web de Recomendaciones de AWS

AWS Servicios de

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

AWS recursos generales

- AWS Documentación de
- Referencia general de AWS
- Glosario de AWS

Historial de documentos

En la siguiente tabla, se describen cambios significativos de esta guía. Si quiere recibir notificaciones de futuras actualizaciones, puede suscribirse a las notificaciones RSS.

Cambio	Descripción	Fecha
Publicación inicial	—	29 de enero de 2021

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.