Apuntes de MLOps

por: Fode

27 de marzo de 2024

Capítulo 1

Conceptos MLOps

1.1. Introducción

1.1.1. ¿Qué es MLOps?

MLOps es la abreviación de Machine Learning Operations y describe el conjunto de prácticas para diseñar, implementar y mantener el aprendizaje automático en producción ed manera continua, confiable y eficiente.

Operaciones de Machine Learning

En producción significa que nos centramos en el aprendizaje automático que se utiliza en los procesos empresariales en lugar de en un modelo de aprendizaje automático que solo existe en su computadora portátil. También analizamos algo más que simplemente entrenar un modelo de aprendizaje automático. MLOps se aplica a lo que llamamos el ciclo de vida del aprendizaje automático, que incluye desde el diseño y desarrollo hasta el mantenimiento del aprendizaje automático en producción.

¿Por qué MLOps?

Imagínese a un chef en un restaurante preparando platos maravillosos para sus invitados. Antes de preparar una receta para un plato nuevo, necesita saber qué ingredientes tiene y qué quieren los invitados. A partir de ahí, desarrolla la mejor receta posible. Para entregar el plato a los invitados, se requieren personas en el servicio y también se necesita equipo para preparar el plato. La combinación de todos estos factores lo convierten en un proceso complejo. Para estructurar el proceso, se podrían realizar controles rutinarios de los equipos, una degustación diaria del plato y se podrían probar los platos con diferentes invitados. Del mismo modo, el desarrollo del aprendizaje automático también es un proceso complejo. **Antes de desarrollar un modelo de aprendizaje automático, debemos pensar en los requisitos comerciales y el valor agregado de nuestro modelo. Combinando los datos y los requisitos, necesitamos encontrar el algoritmo adecuado, la receta.** Durante el desarrollo también necesitamos un ordenador, lo que también plantea sus propios retos, al igual que el equipamiento de la cocina. **MLOps tiene como objetivo estructurar el proceso y mitigar los riesgos involucrados en el desarrollo, implementación y mantenimiento del aprendizaje automático.**

El origen de MLOps

MLOps se origina en Development Operations, también llamado DevOps para abreviar. **DevOps describe un conjunto de prácticas y herramientas que se pueden aplicar al desarrollo de software para garantizar que el software se desarrolle de forma continua, confiable y eficiente**. El desarrollo de software tradicional solía ser lento debido a la separación de los equipos de Desarrollo y Operaciones. El equipo de desarrollo está formado por las personas que escriben el código, que fueron separadas del equipo de operaciones, las personas que implementan y dan soporte al código. Es por eso que DevOps es una integración de ambos equipos. De manera similar a cómo se aplica DevOps al desarrollo de software, MLOps se aplica al desarrollo de aprendizaje automático.

¿Qué operaciones?

Al igual que DevOps y MLOps, también existen mejores prácticas y herramientas para departamentos similares que podemos encontrar en una organización de TI, como ModelOps, DataOps y AIOps. Cada Ops se origina en la misma filosofía de DevOps y se centra en un desarrollo continuo, confiable y eficiente. ModelOps puede verse como una extensión de MLOps, con un conjunto de prácticas centradas principalmente en el modelo de aprendizaje automático. DataOps se centra en las mejores prácticas en materia de calidad y análisis de datos. Dado que los datos son parte del aprendizaje automático, esto se superpone con MLOps. AIOps significa Inteligencia Artificial para Operaciones de TI y es más amplio que el simple aprendizaje automático. En AIOps, se utilizan análisis, big data y aprendizaje automático para resolver problemas de TI sin asistencia o intervención humana.

Beneficios de MLOps

El uso de prácticas y herramientas de MLOps tiene múltiples beneficios. Puede, por ejemplo, mejorar la velocidad general de desarrollo y entrega de modelos de aprendizaje automático. Los procesos también se vuelven más confiables y seguros gracias a MLOps. Inherente a MLOps es que tiene como objetivo cerrar la brecha entre el aprendizaje automático y los equipos de operaciones, lo que mejora la colaboración. Durante este curso, analizaremos cómo MLOps pretende proporcionar estos beneficios.

1.1.2. Diferentes fases en MLOps

Ciclo de vida de MLOps

El ciclo de vida del aprendizaje automático es uno de los conceptos fundamentales en MLOps. Consta de tres fases amplias:

1. Diseño.
2. Desarrollo.
3. Implementación.

Este es un proceso iterativo y cíclico en el que no es raro ir y venir entre fases. Es importante dedicar tiempo a cada fase, ya que todas desempeñan un papel importante en el ciclo de vida completo. **Durante cada fase, es importante evaluar constantemente con las partes interesadas si el proyecto de aprendizaje automático debe continuar.** Podría ser que descubramos durante la fase de diseño que solo tenemos datos limitados o que solo podemos aplicar el problema a un grupo pequeño. Esto reduce el valor añadido y, por tanto, requiere una evaluación adicional por parte de las partes interesadas.

¿Por qué el ciclo de vida del aprendizaje automático?

El ciclo de vida del aprendizaje automático es importante porque brinda una descripción general de alto nivel de cómo se debe estructurar un proyecto de aprendizaje automático para brindar valor real y práctico. También define los roles que se requieren en cada paso para que el proyecto sea un éxito. Como veremos, podemos aplicar ciertas prácticas y herramientas a cada fase para optimizar el ciclo de vida.

Fase de diseño

En la fase de diseño, nos centramos en el diseño del proyecto de aprendizaje automático.

1. Definir el contexto del problema.
2. Determinar el valor añadido del uso del aprendizaje automático.
3. Recopilar requisitos comerciales.
4. Establecer métricas clave. (Para rastrear el progreso del ciclo de vida del aprendizaje automático).
5. Recopilar datos.
6. Asegurarnos de que la calidad de los datos sea suficiente para desarrollar un modelo de aprendizaje automático.

Fase de desarrollo

En la fase de desarrollo, nos centramos en desarrollar el modelo de aprendizaje automático. Hacemos esto experimentando con una combinación de datos, algoritmos e hiperparámetros de acuerdo con el diseño de implementación. Durante el experimento, entrenamos y evaluamos uno o más modelos para encontrar el más adecuado. **El objetivo de la fase de desarrollo es terminar con el modelo de aprendizaje automático más adecuado y listo para su implementación.**

Fase de implementación

En la fase de implementación, integramos el modelo de aprendizaje automático que desarrollamos anteriormente en el proceso comercial. Esto podría implicar la **creación de un microservicio** a partir del modelo de aprendizaje automático. **Un microservicio es una pequeña aplicación que incluye el modelo de aprendizaje automático de modo que podamos integrarlo fácilmente en el proceso de negocio.** También pretendemos establecer un seguimiento del modelo de aprendizaje automático. *Podemos configurar alertas cuando encontremos una desviación de datos o cuando nuestro modelo ya no genere una predicción.* La deriva de datos ocurre cuando nuestros datos cambian, lo que afecta el modelo de aprendizaje automático. Analizaremos estos conceptos con mayor detalle más adelante, donde analizaremos los diferentes componentes de cada fase.

1.1.3. Roles en MLOps

A partir de las fases que hemos visto en el ciclo de vida del aprendizaje automático, podemos crear una secuencia de ejemplo de pasos por los que pasa el ciclo de vida. Para cada tarea se requieren diferentes roles. Analizaremos dos categorías de roles, a saber, roles comerciales y roles técnicos.

Roles comerciales

En la categoría empresarial, podemos distinguir dos roles principales,

- El interesado en la empresa.
- El experto en la materia.

Veamos primero el papel de las partes interesadas en el negocio.

Roles empresariales: partes interesadas del negocio A veces también se hace referencia a la parte interesada del negocio como propietario del producto. Son personal directivo que toma decisiones presupuestarias y se asegura de que el proyecto de aprendizaje automático esté alineado con la visión de alto nivel de la empresa. **Están involucrados durante todo el ciclo de vida.** Primero, definen los requisitos comerciales durante la fase de diseño. En la fase de desarrollo también comprueban si los resultados iniciales de los experimentos son satisfactorios. Más adelante en la fase de implementación, vuelven a examinar si el resultado del ciclo de vida es el esperado.

Roles comerciales: experto en la materia En segundo lugar, está el experto en la materia, que tiene conocimientos de dominio sobre el problema que intentamos resolver. En el comercio minorista, podría ser, por ejemplo, alguien del equipo de ventas que conozca las variables que influyen en las ventas. El experto en la materia participa durante todo el ciclo de vida porque puede ayudar a las funciones más técnicas a interpretar los datos y los resultados en cada paso.

Roles técnicos

Hay cinco roles técnicos principales involucrados en el ciclo de vida del aprendizaje automático:

- Ingeniero de datos.
- Científico de datos.
- Ingeniero de software.
- Ingeniero de aprendizaje automático.
- Ingeniero de backend.

Roles técnicos: ingeniero de datos El ingeniero de datos **es responsable de la recopilación, almacenamiento y procesamiento de datos**. Esto también significa que el ingeniero de datos **debe verificar la calidad de los datos e incluir pruebas para que la calidad se mantenga durante todo el proceso**. Por lo tanto, el ingeniero de datos **participa principalmente en tareas que tienen que ver con los datos antes de entrenar el modelo, durante el entrenamiento del modelo y una vez que el modelo se utiliza en producción**.

Roles técnicos: científico de datos El científico de datos **es responsable del análisis de datos y la capacitación y evaluación de modelos**. La evaluación incluye el seguimiento del modelo una vez que se ha implementado para garantizar que las predicciones del modelo sean válidas. Podemos encontrar al científico de datos en todas las fases del ciclo de vida, pero **principalmente durante la fase de desarrollo**.

Roles técnicos: ingeniero de software El ingeniero de software **participa principalmente en la fase de implementación, donde escribe software para ejecutar el modelo, implementar el modelo y monitorear si el modelo está y permanece en línea una vez implementado**. También se aseguran de que el código esté escrito de acuerdo con pautas comunes. Dado que la implementación es una parte importante del ciclo de vida del aprendizaje automático, el ingeniero de software también debe participar en la fase de diseño.

Roles técnicos: ingeniero de ML El ingeniero de aprendizaje automático es un rol relativamente nuevo que es bastante versátil y **está diseñado específicamente para tener experiencia en todo el ciclo de vida del aprendizaje automático**. Es una función multifuncional que se superpone con las demás funciones técnicas. Como tal, el ingeniero de aprendizaje automático **participa en todas las fases**. Saben, por ejemplo, cómo extraer y almacenar datos y desarrollar o implementar un modelo de aprendizaje automático.

Roles técnicos: ingeniero backend El ingeniero de backend es alguien que **participa principalmente en la configuración de la infraestructura de la nube para permitir el desarrollo y la implementación de modelos de aprendizaje automático**. Podría ser una base de datos para almacenar los datos, pero también las computadoras que ejecutan el modelo de aprendizaje automático.

1.2. Diseño de MLOps

1.2.1. Diseño de aprendizaje automático

Ahora analizaremos el inicio de la fase de diseño, en la que investigamos

1. El valor añadido.
2. Los requisitos comerciales.
3. Las métricas clave.

que debemos rastrear.

Valor añadido

Normalmente, el ciclo de vida del aprendizaje automático comienza con la determinación del valor agregado de crear y ejecutar el modelo de aprendizaje automático. Esto **suele expresarse en términos de dinero o tiempo**. Determinar el valor de un modelo de aprendizaje automático puede ser un poco aproximado, pero es aconsejable estimar el potencial que tiene un determinado proyecto.

Estimación del valor agregado Digamos que tenemos un modelo de aprendizaje automático que predice si un cliente va a abandonar. Tenemos una base de clientes de 100.000 clientes que tienen una suscripción de $10 cada mes. Si podemos predecir esto correctamente para el 80 % de los 1000 clientes que normalmente abandonan, podemos enviarles a estos clientes una suscripción con descuento para que permanezcan el 50 % del tiempo. Esto da como resultado 1000 clientes multiplicados por ochenta por ciento multiplicados por cincuenta por ciento, es decir, 400 clientes

que no abandonan. Si les damos a estos 400 clientes una suscripción con descuento de ocho dólares, podemos ahorrar un total de 3200 por mes.

Requisitos comerciales Aparte del valor añadido del modelo de aprendizaje automático, también debemos considerar los requisitos del negocio. Especialmente en la fase de diseño, **es fundamental pensar en el usuario final del modelo de aprendizaje automático**. Digamos que estamos construyendo un modelo de aprendizaje automático que predice el número de ventas de un producto específico, de modo que podamos comprar la cantidad correcta para poner en nuestra tienda. El modelo de aprendizaje automático generará una cantidad prevista de ventas. Debemos considerar la frecuencia de las predicciones y qué tan rápido las necesitamos. También debemos evaluar la precisión del modelo y si sus resultados son explicables para los no expertos. **La transparencia se está convirtiendo en un factor cada vez más importante en el desarrollo del aprendizaje automático, ya que nos permite descubrir por qué el modelo hace sus predicciones, por qué está equivocado y cómo podemos mejorar el modelo. Todos estos requisitos tienen un impacto en el algoritmo que usaremos**.

Dependiendo del problema que estemos tratando de resolver, **también podrían existir requisitos regulatorios y de cumplimiento para el uso del aprendizaje automático**. Por ejemplo, en finanzas, cuando la ley exige una explicación por parte del sistema. **También debemos tener en cuenta el presupuesto disponible y el tamaño del equipo**.

Métricas clave Para ver si el ciclo de vida del aprendizaje automático avanza según lo esperado, suele ser aconsejable realizar un seguimiento del rendimiento del modelo. Sin embargo, como hemos visto anteriormente, los roles involucrados en los procesos MLOps son multidisciplinarios y, por lo tanto, también tienen su propia forma de rastrear el desempeño. El científico de datos analiza la precisión de un modelo, cuántas veces el algoritmo es correcto.

El experto en la materia está interesado en el impacto del modelo en el negocio, por ejemplo, en cómo mejora su trabajo gracias al uso del aprendizaje automático. Están interesados principalmente en métricas específicas de dominio.

La parte interesada del negocio está más interesada en el valor monetario del modelo, en cuántos casos realmente generamos ingresos. Esto suele expresarse en dinero o tiempo. Para aprovechar al máximo el aprendizaje automático, debemos alinear las diferentes métricas para asegurarnos de que todos estén en la misma página.

1.2.2. Calidad e ingesta de datos

La recopilación de datos es parte de la fase de diseño. Durante esta fase, investigamos:

- La calidad de los datos.
- Cómo extraemos los datos requeridos.

¿Qué es la calidad de los datos?

Según DAMA-DMBOK, un marco para la gestión de datos, el término **calidad de los datos se refiere tanto a las características asociadas con datos de alta calidad como a los procesos utilizados para medir o mejorar la calidad de los datos**. La calidad de un modelo de aprendizaje automático depende en gran medida de la calidad de los datos. Los datos son el núcleo del modelo de aprendizaje automático. Por lo tanto, tener una visión clara de la calidad de los datos es crucial para el éxito del ciclo de vida del aprendizaje automático. Tener una mala calidad de los datos es perjudicial para el rendimiento del modelo de aprendizaje automático. **Mejorar la calidad de los datos suele ser el primer paso para mejorar el rendimiento del modelo.**

1.2.3. Dimensiones de la calidad de los datos

La calidad de los datos se puede definir según cuatro dimensiones principales:

- Precisión.- describe el grado en que los datos son exactos o correctos para la tarea en cuestión.
- Exhaustividad.- se refiere al grado en que los datos describen completamente el problema en cuestión.

- Coherencia.- Un departamento podría tener una definición diferente de lo que constituye un cliente activo que otro, lo que hace que los datos sean inconsistentes.
- Puntualidad.- se refiere al plazo en el que los datos estarán disponibles.

Ejemplo de dimensiones de calidad de datos Digamos que estamos construyendo un modelo predictivo para determinar si los clientes abandonarán. Podemos comprobar la calidad de los datos de los clientes repasando las cuatro dimensiones. Un ejemplo de precisión sería si los datos describen correctamente al cliente. Podría ser que los datos indiquen que un cliente tiene 18 años, pero en realidad tiene 32 años. Eso sería inexacto. Para estar completo, nos fijamos principalmente en los datos que faltan, por ejemplo, si nos faltan los apellidos de los clientes. Con coherencia, investigamos si la definición de cliente es coherente en toda la organización. Podría ser que un departamento tenga una definición diferente de cliente activo que otro, lo que hace que los datos sean inconsistentes. Si nos fijamos en la puntualidad, nos interesa la disponibilidad de datos. Por ejemplo, cuando los pedidos de los clientes se sincronizan diariamente, no están disponibles en tiempo real. Tener una calidad de datos inferior en una o más dimensiones no significa que el proyecto esté destinado al fracaso. Hay varias cosas que podemos hacer para abordar la baja calidad de los datos, como recopilar más datos o completar los datos faltantes con otras fuentes.

Ingestión de datos

Normalmente, en la fase de diseño, también analizamos cómo extraer y procesar datos. Esto se hace mediante el uso de un canal de datos automatizado, del cual vemos un ejemplo de alto nivel aquí. Una canalización de datos suele ser una parte del ciclo de vida del aprendizaje automático a través del cual los datos se procesan automáticamente. **Un tipo común de proceso de ingesta de datos es ETL, que significa extraer, transformar y cargar**. Describe los tres pasos seguidos en un proceso ETL. Los datos se extraen de la fuente, se transforman al formato requerido y se cargan en alguna base de datos interna o propietaria. En un proceso ETL, también podemos incluir verificaciones automatizadas, como las expectativas que tenemos sobre ciertas columnas de datos. Por ejemplo, esperamos que la columna de temperatura siempre contenga un número. Incluir estas comprobaciones automatizadas en un proceso de datos ayuda a acelerar la fase de desarrollo e implementación del ciclo de vida, ya que los datos defectuosos o de baja calidad afectarán el modelo de aprendizaje automático.

1.2.4. Ingeniería de funciones y tienda de funciones

Ingeniería de funciones

Después de la fase de diseño, la ingeniería de funciones es el siguiente paso en el proceso de desarrollo del aprendizaje automático. La ingeniería de funciones es un componente clave del desarrollo del aprendizaje automático.

La **ingeniería de características es el proceso de seleccionar, manipular y transformar datos sin procesar en características**. Una característica es una variable, como una columna de una tabla. La forma en que creamos funciones a partir de los datos es una parte importante del desarrollo del aprendizaje automático. Podemos usar las funciones tal como aparecen en los datos sin procesar, pero también podemos crear las nuestras propias. Veamos un ejemplo.

Datos del cliente Por ejemplo, tenemos un conjunto de clientes y cierta información sobre sus pedidos. El número de pedidos y el gasto total de cada cliente son dos características diferentes. Para crear una nueva función, podríamos calcular el gasto promedio por cliente. De esta manera, hemos diseñado una nueva característica. Además de los datos de pedido en este ejemplo, a menudo hay muchos más datos disponibles en proyectos reales de aprendizaje automático y, por lo tanto, muchas posibilidades para diseñar funciones. Por lo tanto, la ingeniería de funciones es una parte importante del desarrollo del aprendizaje automático.

Evaluación de ingeniería de funciones

Una ponderación importante en la ingeniería de funciones es cuándo mantener la ingeniería o cuándo detenerla. Realizar una ingeniería de características integral puede producir un modelo muy preciso o lograr más estabilidad. Sin embargo, realizar una ingeniería integral de funciones también tiene un costo, lo que puede afectar el éxito de nuestro proyecto de aprendizaje automático.

- Más funciones pueden ser más costosas, ya que esto puede requerir costosos pasos de preprocesamiento.
- Más funciones también requieren más mantenimiento.
- Más funciones también pueden generar ruido o ingeniería excesiva.

¿Qué pasa si aumenta la cantidad de proyectos de ML?

Imagine que un científico de datos está trabajando en un nuevo proyecto de aprendizaje automático. Para el primer proyecto, podrían simplemente realizar la ingeniería de funciones una vez y entrenar el modelo. Pero, ¿qué pasa si crece la cantidad de proyectos de aprendizaje automático? **A medida que crece el número de proyectos y, por tanto, el número de modelos de aprendizaje automático, almacenar funciones de forma estructurada y centralizada puede acelerar enormemente el desarrollo de modelos de aprendizaje automático. Para hacer esto, una herramienta importante en MLOps es la tienda de funciones. Un almacén de características, como su nombre lo indica, es una herramienta para almacenar características o variables de uso común relevantes para el modelo de aprendizaje automático.**

La tienda de funciones

La tienda de funciones es el lugar central donde se pueden administrar las funciones. Al utilizar un almacén de funciones, un científico de datos puede encontrar las funciones adecuadas para su proyecto, definir nuevas funciones y utilizarlas para entrenar el modelo. También es el lugar donde se pueden monitorear las funciones. Al mismo tiempo, al utilizar un almacén de funciones, nos aseguramos de que las funciones estén listas para usarse como entrada para el modelo de aprendizaje automático en producción cuando lleguen nuevas muestras.

¿Cuándo utilizar una tienda de funciones? No tenemos que utilizar una tienda de funciones al desarrollar un modelo de aprendizaje automático. En algunos casos, puede resultar redundante utilizar una tienda de funciones. Los factores a considerar cuando decide utilizar una tienda de funciones son el costo computacional de las funciones. A veces, las funciones estarán listas como entrada para el modelo de aprendizaje automático tal como están. El uso de una tienda de funciones también depende de la cantidad de proyectos que tengamos. Las respuestas a estas preguntas determinarán si el desarrollo actual del aprendizaje automático se beneficia de una tienda de funciones o no.

1.2.5. Seguimiento de experimentos

Ahora analizaremos los experimentos de aprendizaje automático, qué es el seguimiento de experimentos y por qué es un concepto importante en MLOps.

El experimento de aprendizaje automático

Parte del desarrollo del modelo de aprendizaje automático consiste en realizar experimentos de aprendizaje automático. En un experimento de aprendizaje automático, entrenamos y evaluamos múltiples modelos de aprendizaje automático para encontrar el mejor. Como en cualquier experimento, probamos diferentes configuraciones para ver cuál funciona mejor.

¿Por qué es importante el seguimiento de experimentos?

Durante los experimentos de aprendizaje automático, podemos configurar diferentes modelos de aprendizaje automático, por ejemplo, regresión lineal o redes neuronales profundas. Podemos alterar los hiperparámetros del modelo, como el número de capas en una red neuronal. Podríamos utilizar diferentes versiones de los datos y diferentes scripts para ejecutar el experimento. También podemos usar diferentes archivos de configuración de entorno por experimento, como qué versión de Python o R se usa y qué bibliotecas. Al alterar cada uno de estos factores durante los experimentos, la cantidad de configuraciones diferentes puede volverse enorme. **Cada experimento también tiene un resultado diferente. Por eso es una buena idea realizar un seguimiento de las configuraciones y los resultados de cada experimento.**

Uso del seguimiento de experimentos en el ciclo de vida del aprendizaje automático

Además de rastrear todas las diferentes configuraciones de experimentos, el seguimiento de experimentos puede ayudar a comparar y evaluar experimentos, reproducir resultados de experimentos anteriores, colaborar con desarrolladores y partes interesadas e informar sobre los resultados a las partes interesadas.

¿Cómo realizar un seguimiento de los experimentos? Dependiendo de la madurez de nuestro desarrollo de aprendizaje automático, existen diferentes opciones para realizar un seguimiento de los experimentos. Podríamos empezar utilizando una hoja de cálculo en Excel, donde anotamos los detalles de cada experimento en cada fila. Si hacemos muchos experimentos, esto requerirá mucho trabajo manual. También podríamos crear nuestra propia plataforma de experimentos que rastree automáticamente el experimento durante el entrenamiento del modelo. Tener nuestra propia plataforma experimental nos permite crear una solución personalizada para nuestro proceso específico. Sin embargo, esto puede costar tiempo y esfuerzo. Esto se puede resolver mediante el uso de herramientas modernas de seguimiento de experimentos, ya que están diseñadas para la mayoría de los casos de uso de seguimiento de experimentos. Tenga en cuenta que esto puede resultar costoso. Los resultados del seguimiento del experimento se pueden almacenar en un almacén de metadatos.

Un experimento de aprendizaje automático y el preceso Imaginemos que estamos desarrollando un clasificador que clasifica si hay un perro o un gato en una imagen. Podemos comenzar el primer experimento entrenando una red neuronal con una capa oculta en mil imágenes. En el segundo experimento, incluimos cachorros y gatitos, aumentamos el conjunto de datos a dos mil imágenes y entrenamos una red neuronal con dos capas ocultas.

Si ejecutamos el experimento en el ejemplo de clasificar un perro o un gato, seguiremos los siguientes pasos.

1. Formulamos una hipótesis basada en los objetivos comerciales.
2. Recopilamos los datos necesarios
3. Definimos los hiperparámetros iniciales que deseamos probar.
4. Configuramos el seguimiento del experimento.
5. Ejecutamos el primer entrenamiento del modelo.
6. Durante el entrenamiento del modelo, ajustamos el modelo de aprendizaje automático al conjunto de datos que configuramos para el entrenamiento durante ese experimento
7. Evaluamos los resultados probando el modelo.
8. Registramos el modelo.
9. Visualizamos e informamos estos resultados al equipo o a las partes interesadas para determinar los próximos pasos.

1.2.6. Preparar el modelo para la implementación

Ahora que hemos analizado el diseño y desarrollo del ciclo de vida del aprendizaje automático, es hora de analizar la última fase: la implementación.

Entorno de ejecución

Durante la fase de desarrollo, los científicos de datos utilizaron datos de entrenamiento para desarrollar un modelo de aprendizaje automático.

Del desarrollo al despliegue

El desarrollo se lleva a cabo en el entorno de desarrollo, que suele ser el ordenador local de un científico de datos o un ordenador virtual que se puede controlar de forma remota, por ejemplo, en la nube. Una vez que se desarrolla el modelo de aprendizaje automático, debemos trasladarlo al entorno de producción. En el entorno de producción, el modelo de aprendizaje automático hará predicciones basadas en datos entrantes reales. Una vez implementado, el modelo está activo

y creará un impacto empresarial real. Sin embargo, implementar un modelo en el entorno de producción no es tan sencillo porque están configurados en diferentes entornos de ejecución. Un entorno de ejecución es el entorno en el que se ejecuta el modelo. Durante el desarrollo, el modelo se ejecuta en el entorno de ejecución de desarrollo.

Trabajar en diferentes entornos de ejecución es similar a trabajar en diferentes cocinas. Si creamos una receta en una cocina, los resultados de la misma receta pueden ser muy diferentes en otra cocina. Hay un juego diferente de sartenes, tal vez una cocina tenga electrodomésticos diferentes o un grupo diferente de chefs esté a cargo de la cocina. Todo esto puede afectar al plato que estemos preparando. De manera similar, en un entorno de ejecución de desarrollo, podemos usar diferentes versiones de Python y ciertas bibliotecas. Esto causa problemas porque es posible que el mismo código no funcione en diferentes entornos de ejecución o produzca resultados diferentes.

Mitigar diferentes entornos

Para mitigar el hecho de tener diferentes entornos, podemos utilizar máquinas separadas pero configuradas de forma idéntica. Esto es similar a una computadora normal. Tiene el hardware físico, el sistema operativo, las bibliotecas y la aplicación. Las bibliotecas y la aplicación contienen el entorno de producción y el modelo de aprendizaje automático. Esta es una solución sencilla, pero difícil de mantener y no escalable. Con cada actualización, tenemos que actualizar toda la máquina.

También podemos usar una o más máquinas virtuales en una máquina separada. Cada máquina virtual es como una versión virtual de una computadora física normal con un sistema operativo, bibliotecas y la aplicación. Una computadora que ejecuta máquinas virtuales tiene un hipervisor. Esto ayuda a distribuir los recursos, el hardware de la computadora, entre diferentes máquinas virtuales. Esto es más fácil de mantener pero requiere muchos recursos, porque necesitamos una computadora virtual para cada aplicación.

Por último, podríamos usar algo llamado contenedor. Esto nos permite ejecutar múltiples aplicaciones en una máquina. Los contenedores utilizan menos recursos que una máquina virtual y son más portátiles que las aplicaciones en una máquina virtual. Puede verse como una versión más ligera de la máquina virtual. **La implementación del modelo de aprendizaje automático como contenedor es actualmente el estándar para MLOps**.

Contenedores de beneficios

El uso de contenedores proporciona numerosos beneficios.

- Son más fáciles de mantener.
- Son muy portátiles.
- Se inician rápidamente.

Sin embargo, estos beneficios no significan que cada aplicación deba implementarse como un contenedor. Si la aplicación ha funcionado bien en una máquina virtual y no sufre problemas de diferentes entornos, está bien no utilizar contenedores.

1.2.7. Arquitectura de implementación de aprendizaje automático

Arquitectura de microservicios

Una vez que nos hayamos ocupado de los diferentes entornos de ejecución, debemos pensar en cómo implementar el modelo de aprendizaje automático. Esto también implica que debemos pensar en cómo configuramos la arquitectura.

Arquitectura monolítica frente a microservicio

Digamos que tenemos una tienda web con un servicio de pago, un servicio para el carrito de compras y un servicio para el inventario. Podemos ejecutar cada servicio en el mismo ordenador, también conocido como instancia única. En el desarrollo de software tradicional, las aplicaciones a menudo se creaban en una arquitectura monolítica. Esto significa que la aplicación es una aplicación uniforme que incluye todos los servicios.

Una solución diferente es la arquitectura de microservicios. La arquitectura de microservicios es, a diferencia de la arquitectura monolítica, una colección de servicios más pequeños que se pueden implementar de forma independiente. Si una aplicación falla en una arquitectura de microservicios,

solo falla el servicio por separado, mientras que en una arquitectura monolítica, fallará toda la aplicación. Una aplicación monolítica puede volverse compleja ya que todos los servicios están entrelazados y no son independientes. Esto también hace que sea más difícil escalar. El uso de una arquitectura de microservicios depende de la aplicación. Si nuestra aplicación es muy pequeña, tener un microservicio separado para cada parte aún más pequeña puede resultar costoso porque cada servicio requiere potencia de cálculo y debe mantenerse de forma independiente.

Inferencia

Es una práctica común implementar el modelo de aprendizaje automático como un microservicio. Esto nos permite utilizar el modelo de aprendizaje automático para hacer predicciones basadas en datos nuevos e invisibles.

Este proceso también se llama inferencia. Es el proceso en el que enviamos nuevos datos, por ejemplo, los datos de entrada de un cliente, para los cuales el modelo de aprendizaje automático inferirá un resultado. En este caso, el resultado es una predicción que contiene la probabilidad de que un cliente abandone.

Interfaz de programación de aplicaciones (API)

Para proporcionar comunicación entre microservicios, se utiliza una interfaz de programación de aplicaciones (o API). Una API es un conjunto de combinaciones de entrada y salida predefinidas que permite que diferentes servicios se comuniquen. Es similar al menú de un restaurante. Si se pide una pizza carciofo usando un menú, la cocina sabe que necesita juntar la base de la pizza, la berenjena, etcétera, y lo entregará una vez hecho para que el servicio pueda entregárselo al huésped. La entrada es el pedido en el menú, que se envía a la cocina, el modelo de aprendizaje automático, que devuelve el artículo pedido. No tener una API sería como no tener un menú y, como tal, no tendríamos una comunicación adecuada entre los servicios.

Flujo de datos API

1. Llegan datos de entrada nuevos e invisibles.
2. Los datos de entrada se envían a la API.
3. La API envía datos de entrada al modelo de aprendizaje automático.
4. El modelo de aprendizjae automático hace una predicción basada en nuevos datos de entrada.
5. La salida del modelo se envía de vuelta a la API.
6. La API comunica las predicciones del modelo a la aplicación.

Integración

Una vez que el modelo se ha implementado como un microservicio y la API nos permite inferir el modelo, se requiere un último paso. El último paso es integrar el modelo dentro del proceso de negocio. Esto es diferente para cada negocio, pero la mayoría de las veces implica conectar la API con el sistema que ya está implementado. Antes de utilizar el modelo de aprendizaje automático en producción, es una práctica común probar primero el modelo con una muestra de los datos para asegurarnos de que todo funcione como se esperaba.

1.2.8. CI/DI y estrategia de implementación

El proceso de CI/CD también forma parte de la fase de implementación.

CI/CD

El uso de integración continua e implementación continua (o CI/CD) es un concepto importante dentro del desarrollo de software. CI/CD se originó en DevOps y **se centra en automatizar la implementación de código. Es una serie de pasos para desarrollar, probar e implementar el código**. Al utilizar una canalización de CI/CD, los desarrolladores de software pueden realizar fácilmente cambios incrementales y luego impulsar estos cambios al entorno de producción. Estos mismos principios se pueden aplicar al desarrollo e implementación de código para modelos de aprendizaje automático.

Integración continua CI **La integración continua es la práctica en la que los cambios de código se integran continuamente de forma rápida y frecuente**. Cada cambio se prueba automáticamente cuando se confirman y fusionan. De esta manera, podemos identificar errores y fallos fácilmente y asegurarnos de que muchos desarrolladores puedan trabajar juntos en el mismo código.

Despliegue continuo CD La implementación continua funciona junto con la integración continua al automatizar la publicación del código que se validó durante el proceso de integración continua. **El objetivo de la práctica de la implementación continua es tener siempre código listo para producción**.

Canalización de CI/CD

Configurar una canalización de CI/CD puede resultar tedioso al principio, pero puede acelerar enormemente el proceso de implementación. Es como tener una lista de comprobaciones a realizar antes del lanzamiento de una nueva receta en un restaurante. Podemos crear fácilmente una nueva receta, realizar cambios en la receta y comprobar si cumple con los procesos actuales. En resumen, la integración continua es un conjunto de prácticas mientras se escribe el código para ejecutar el modelo de aprendizaje automático. La implementación continua es un conjunto de prácticas una vez completado el código.

Estrategias de implementación

Una vez que un modelo de aprendizaje automático está listo para implementarse, podemos elegir diferentes estrategias de implementación. Cada estrategia tiene una forma diferente de reemplazar el antiguo modelo de aprendizaje automático por el nuevo modelo de aprendizaje automático. Analizaremos tres estrategias de implementación:

- Implementación básica.- simplemente reemplazamos el modelo antiguo con el nuevo modelo en producción. Todos los datos de entrada nuevos se enviarán al nuevo modelo en lugar del modelo anterior.

- Implementación paralela.- enviamos nuevos datos tanto al modelo nuevo como al modelo anterior. Todavía utilizamos el modelo antiguo en producción. Se probará el resultado de ambos modelos para garantizar que el nuevo modelo funcione como se espera.

- Implementación canaria.- utilizamos el nuevo modelo en producción, pero solo para una pequeña parte de los nuevos datos entrantes. De esta manera, utilizamos el nuevo modelo de inmediato, pero en caso de que falle, solo un pequeño número de usuarios se verá afectado.

La implementación básica es la más fácil de implementar y utiliza la menor cantidad de recursos porque el nuevo modelo reemplaza completamente al anterior. Esto conlleva un alto riesgo en caso de que el modelo no funcione como se esperaba. La implementación paralela es similar en términos de implementación, pero utiliza más recursos ya que ejecutamos ambos modelos por completo en lugar de reemplazar uno por el otro. No hay riesgo cuando el modelo no funciona como se esperaba. La implementación canary es un poco más difícil de implementar pero utiliza menos recursos que tener dos modelos completamente implementados. Sin embargo, el riesgo es ligeramente mayor cuando el nuevo modelo no funciona como se esperaba.

1.2.9. Automatización y escalado

Hasta ahora, hemos analizado los componentes del diseño, desarrollo e implementación del aprendizaje automático. El ciclo de vida del aprendizaje automático es un proceso experimental, lo que significa que con frecuencia tenemos que ir y venir por las diferentes fases. Por tanto, la automatización puede ayudar enormemente a acelerar el ciclo de vida. Por ejemplo, nos permite repetir fácilmente los mismos experimentos varias veces.

Dado que el aprendizaje automático suele trabajar con grandes cantidades de datos, también es necesario configurar un sistema escalable. Por lo tanto, la automatización y el escalado son conceptos cruciales en MLOps.

Fase de diseño

La fase de diseño es la fase más importante dentro del ciclo de vida del aprendizaje automático. Sin un objetivo decente y datos de alta calidad, las otras dos fases podrían fracasar. Dado que el aprendizaje automático es multidisciplinario, como hemos visto por los diferentes roles involucrados, es importante que todos estén alineados. En términos de automatización y escalamiento, la fase de diseño sigue siendo un proceso manual. Sin embargo, el diseño se puede modelar para obtener el valor agregado, los requisitos comerciales y las métricas clave. Esto convierte la fase de diseño en un proceso estructurado en línea con las prácticas de MLOps. La adquisición de datos y los controles de calidad de los datos se pueden automatizar. Dado que la calidad del modelo de aprendizaje automático depende de la calidad de los datos, la automatización del proceso de adquisición de datos mejora las posibilidades de utilizar con éxito el aprendizaje automático en producción.

Fase de desarrollo

En la fase de desarrollo, utilizamos una tienda de funciones para rastrear y desarrollar funciones. Un almacén de funciones ahorra tiempo que se habría dedicado a crear las mismas funciones utilizadas en experimentos anteriores. Utilizamos el seguimiento de experimentos para automatizar el seguimiento del desarrollo del aprendizaje automático. Esto también ayuda a evaluar los modelos y alinearlos con las métricas clave establecidas en la fase de diseño. El seguimiento del experimento también garantiza que el proceso de desarrollo sea reproducible. Podemos encontrar qué configuraciones se utilizaron y cuáles fueron los resultados.

Fase de implementación

En la fase de implementación, podemos utilizar la contenedorización para mitigar diferentes entornos de ejecución. En términos de escalabilidad, tener aplicaciones en contenedores facilita el inicio de múltiples versiones de la misma aplicación cuando llegan más solicitudes. Por ejemplo, cuando la empresa crece y necesitamos predecir la pérdida de clientes para muchos clientes al mismo tiempo. Se utiliza una canalización de CI/CD para permitir cambios incrementales rápidos durante el desarrollo mediante el uso de la automatización. Esto permite que varios desarrolladores trabajen en el mismo código y ayuda a automatizar el proceso de desarrollo e implementación.

Una arquitectura de microservicios puede ser de gran ayuda a la hora de escalar el aprendizaje automático. Cada nuevo servicio se puede desarrollar e integrar de forma independiente sin afectar a otros servicios.

1.3. Mantenimiento del aprendizaje automático en producción

1.3.1. Monitoreo de modelos de aprendizaje automático

Seguimiento y reciclaje

El seguimiento y el reentrenamiento de los modelos de aprendizaje automático es la última parte de la fase de implementación. Primero examinaremos el seguimiento.

Monitoreo

Cuando un modelo de aprendizaje automático se implementa en producción, todavía no hemos terminado. En producción, el modelo de aprendizaje automático comenzará a hacer predicciones basadas en datos nuevos e invisibles. Para asegurarnos de que el modelo funcione como se esperaba, debemos monitorearlo.

Tipos de seguimiento

Podemos monitorear el modelo observando los datos de entrada y la salida del modelo, sus predicciones. A esto se le llama seguimiento estadístico. Por ejemplo, **podríamos monitorear la probabilidad prevista de que un cliente abandone.**

También podemos analizar métricas más técnicas del modelo. A esto se le llama monitoreo computacional. **Podría ser la cantidad de solicitudes entrantes que se realizan, el uso de**

la red del modelo o la cantidad de recursos que utiliza un servidor para mantener el modelo en ejecución.

Seguimiento estadístico y computacional Podemos verlo de esta manera. Podemos controlar la cocina en la que estamos cocinando de dos formas. En primer lugar, podemos controlar si todos los electrodomésticos siguen funcionando, si el gas y la electricidad están encendidos y si hay gente trabajando en la cocina. Todo lo que no tenga que ver con la comida. Esto sería un seguimiento computacional. En segundo lugar, podemos monitorizar las entradas y salidas de la cocina en cuanto a lo que se trata, la comida. Cuál es la calidad de los ingredientes que entran en la cocina y si el sabor de los platos que salen de la cocina está bien. A esto se le llama seguimiento estadístico.

Bucle de retroalimentación

Con el tiempo, descubriremos si ese cliente realmente ha abandonado. El resultado real también se conoce como verdad fundamental. Utilizando la verdad básica, podemos averiguar si el modelo funciona como se esperaba o si la calidad del modelo se deterioró con el tiempo. Este ciclo en el que comparamos el resultado del modelo con la verdad fundamental se llama ciclo de retroalimentación. El circuito de retroalimentación es una parte crucial para mejorar el modelo de aprendizaje automático. Utilizando el circuito de retroalimentación, podemos descubrir cuándo y por qué el modelo estaba equivocado. Podríamos, por ejemplo, ver que el modelo hace una predicción errónea para grupos de clientes concretos.

Es aconsejable monitorear las métricas tanto estadísticas como computacionales. Esto ayudará a ver dónde podría tener problemas el modelo de aprendizaje automático y nos permitirá mitigar esos problemas, que analizaremos en la siguiente lección.

1.3.2. Entrenando a un modelo de machine learning

Reciclaje después de los cambios

Lo inherente a los datos es que cambian con el tiempo. Es un hecho que el mundo está cambiando y, **dado que nuestro modelo de aprendizaje automático depende de los datos, estos cambios también afectan el modelo. Esta es también la razón por la que un modelo podría necesitar reentrenamiento**. Reentrenamiento significa que utilizamos nuevos datos para desarrollar una nueva versión del modelo de aprendizaje automático, de modo que aprenda y se ajuste a nuevos patrones.

Desviación de los datos

En un problema típico de aprendizaje automático, tenemos datos de entrada y datos de salida, que también se conocen como variable objetivo. Los datos de entrada son las variables utilizadas para predecir la variable objetivo. Si analizamos el caso de predecir si un cliente abandonará, tendremos datos sobre el cliente, que son los datos de entrada. La variable objetivo, en este caso, es si el cliente abandonará, representado por los números cero, no abandonó y uno abandonó. Hay dos cambios principales posibles en este tipo de conjunto de datos:

- Deriva de datos.- un cambio en los datos de entrada.
- Deriva de conceptos.- un cambio en la relación entre los datos de entrada y la variable objetivo. por ejemplo, cuando los mismos datos de entrada hacen que un cliente no abandone en lugar de abandonar. En ese caso, la relación entre los datos de entrada y salida ha cambiado. La deriva del concepto podría hacer que el rendimiento de nuestro modelo se deteriore porque los patrones en los que se entrenó previamente el modelo ya no se mantienen.

¿Con qué frecuencia volver a capacitarse?

La frecuencia con la que se debe volver a entrenar depende de varios factores. El primero es el entorno empresarial. Un entorno empresarial puede estar más sujeto a cambios que otros. Esto también puede ser identificado por un experto en la materia que tenga más conocimiento sobre el medio ambiente, por ejemplo, cuándo podría esperar un cambio. En segundo lugar, la frecuencia con la que se debe volver a capacitar también depende del costo de la misma. Entrenar un modelo requiere recursos. Dependiendo de la complejidad del modelo, el reciclaje requiere más recursos y, por tanto, más dinero. Por último, los requisitos comerciales influyen en la frecuencia con la que se

vuelve a entrenar el modelo. Si se requiere que el modelo tenga siempre una precisión superior al 90 % y un pequeño cambio en los datos hace que la precisión disminuya por debajo de ese umbral, será necesario volver a entrenar el modelo con más frecuencia. **La rapidez con la que disminuye la precisión del modelo también se denomina degradación del modelo.**

Métodos de reciclaje

Cuando volvemos a entrenar, se obtiene un nuevo modelo utilizando nuevos datos. Podríamos usar un modelo que solo use datos nuevos, de modo que haya un modelo separado entrenado con datos antiguos y un modelo entrenado con datos nuevos. También podríamos combinar datos nuevos y antiguos para desarrollar un nuevo modelo. Esto también dependerá del dominio, el costo y el rendimiento del modelo requerido.

Reentrenamiento automático

Dependiendo de la madurez del aprendizaje automático dentro de la empresa, también podríamos aplicar un reentrenamiento automático una vez que se detecte una cierta cantidad de datos o una deriva de concepto. Por ejemplo, cuando detectamos que la edad media de los clientes está cambiando.

1.3.3. Niveles de madurez de MLOps

Madurez de MLOps

Podemos observar el ciclo de vida del aprendizaje automático y determinar qué tan maduras son sus prácticas MLOps. La madurez de MLOps **tiene que ver con la automatización, la colaboración y el monitoreo dentro de los procesos de operaciones y aprendizaje automático en una empresa**. No significa necesariamente que un mayor nivel de madurez de MLOps sea mejor. Sin embargo, **muestra dónde existe potencial de mejora para permitir aún más el uso del aprendizaje automático dentro de la empresa**. Los niveles se aplican principalmente a la fase de desarrollo e implementación. La fase de diseño no se puede automatizar completamente ya que requiere participación humana de múltiples roles diferentes, pero se pueden usar plantillas para que la fase avance más rápida y suavemente.

Podemos distinguir tres niveles. Cada uno con su propio nivel de automatización, colaboración y monitoreo.

1. Procesos manuales.
2. Desarrollo automatizado.
3. Desarrollo e implementación automatizados.

En el nivel 1, no hay ninguna automatización y los equipos de operaciones y aprendizaje automático trabajan de forma aislada. En el nivel 2, hay automatización en el desarrollo de modelos de aprendizaje automático, y los equipos de operaciones y aprendizaje automático colaboran juntos cuando un nuevo modelo está listo para su implementación. En el nivel 3, el ciclo de vida del aprendizaje automático está completamente automatizado durante las fases de desarrollo e implementación.

Nivel 1: Procesos manuales En el nivel más bajo de madurez de MLOps, no existen procesos automatizados. Desde la ingesta de datos hasta la implementación del modelo, todo debe hacerse manualmente. Los equipos o roles que trabajan en el caso de uso lo hacen de forma aislada. Cada fase pasa a la siguiente y hay poca colaboración. Hay poca o ninguna trazabilidad. No se realiza un seguimiento de las funciones utilizadas, los experimentos y el rendimiento del modelo. Una empresa que acaba de empezar a utilizar el aprendizaje automático comenzará en este nivel. Dado que todos los procesos son manuales, el desarrollo y la implementación llevarán más tiempo e implicarán más trabajo, especialmente cuando algo sale mal durante una de las fases.

Nivel 2: Desarrollo automatizado En el segundo nivel de madurez de MLOps, ya no todos los procesos son manuales. Existe automatización en el proceso de desarrollo del modelo de aprendizaje automático. Por lo general, esto se hace mediante el uso de tiendas de funciones y capacitación de modelos automatizada. Existe un proceso de integración continua, pero una vez desarrollados, los modelos aún no se implementan automáticamente. Existe cierta colaboración entre los equipos de operaciones y aprendizaje automático. Sin embargo, la implementación de nuevos modelos

todavía se realiza de forma manual. Existe cierta trazabilidad en este nivel, especialmente durante el proceso de desarrollo. Es fácil reproducir modelos y realizar un seguimiento del rendimiento del modelo durante el desarrollo. Después de la implementación, suele haber una pequeña cantidad de seguimiento.

Nivel 3: Desarrollo e implementación automatizados En el nivel más alto de madurez de MLOps, el desarrollo y la implementación de modelos de aprendizaje automático están automatizados. Existe un proceso completo de CI/CD para desarrollar, probar e implementar nuevos modelos en producción. Existe una estrecha colaboración entre los diferentes roles involucrados en el proceso de aprendizaje automático. Los modelos de aprendizaje automático en producción se monitorean y, en algunos casos, incluso se activan automáticamente para que se vuelvan a entrenar.

1.3.4. Herramientas MLOps

Desde la aparición de MLOps, se han desarrollado muchas herramientas que pueden mejorar la eficiencia y confiabilidad de los procesos de aprendizaje automático. Algunas de estas herramientas son incluso de código abierto. Profundicemos en algunas posibles herramientas que podemos usar por componente analizado a lo largo de este curso. `https://www.datacamp.com/blog/infographic-data-and-machine-learning-tools-landscape`

Tienda de funciones

Para la tienda de funciones, hay varias herramientas disponibles, como Feast y Hopsworks. Feast es una tienda de funciones de código abierto; el nombre es un acrónimo de Feature and Store. Feast es una tienda de funciones autoadministrada, lo que significa que tenemos que administrarla nosotros mismos, lo que requiere más trabajo pero también proporciona más flexibilidad en comparación con otras tiendas de funciones. Hopsworks también es una tienda de funciones de código abierto, parte de la plataforma más grande Hopsworks. Por lo tanto, es más probable que se utilice si el resto de herramientas de Hopsworks ya están en uso.

4. Seguimiento de experimentos 00:58 - 01:23 Para el seguimiento de experimentos, podemos utilizar MLFlow, ClearML y Weights and Biases, entre otros. MLFlow y ClearML ofrecen herramientas para el ciclo de vida del aprendizaje automático, incluido el seguimiento de experimentos. MLflow se especializa en el desarrollo de aprendizaje automático, mientras que ClearML también proporciona herramientas para implementar modelos. Weights and Biases se centra principalmente en rastrear y visualizar los resultados de los experimentos.

Contenedorización

Para la contenedorización, Docker es la herramienta más popular para contener una aplicación. Kubernetes se utiliza para ejecutar la aplicación en contenedores, lo que permite la implementación y escalabilidad automáticas. Además de estas herramientas de código abierto, los proveedores de nube AWS, Azure y Google Cloud también ofrecen sus propias herramientas para ejecutar aplicaciones en contenedores.

Canalización de CI/CD

Para proporcionar canales completos de CI/CD existen herramientas como Jenkins y GitLab. Jenkins es una herramienta de CI/CD de código abierto, mientras que GitLab no lo es. Ambas herramientas permiten a los desarrolladores trabajar juntos en el código mediante un repositorio. Para cada proyecto, suele haber un repositorio independiente, que podemos ver como un directorio que contiene todo el código del proyecto.

Monitoreo

Existe una amplia gama de herramientas para monitorear proyectos de aprendizaje automático. Podemos distinguir herramientas que se centran en el seguimiento del modelo de aprendizaje automático y herramientas que monitorean los datos. Tanto Fiddler como Great Expectations proporcionan herramientas de seguimiento estadístico. Fiddler se centra en el rendimiento del modelo, por ejemplo, qué tan bien están funcionando las predicciones de nuestro modelo. Great Expectations se centra en el seguimiento de datos, por ejemplo, cuántos datos faltan en una determinada columna.

Plataformas MLOps

También hay herramientas disponibles que proporcionan una plataforma completa del ciclo de vida del aprendizaje automático. Cada proveedor de nube, AWS, Azure y Google, tiene uno. Se llaman AWS Sagemaker, Azure Machine Learning y Google Cloud AI Platform. Las herramientas que abarcan todo el ciclo de vida del aprendizaje automático proporcionan herramientas para cada tarea del ciclo de vida. Esta podría ser una herramienta para realizar exploración y procesamiento de datos, pero también un almacén de características y una herramienta de capacitación de modelos.