

Herramientas para la Computación en la Nube Dirigida a
Inteligencia Artificial

Tema 4. Creación de modelos de IA en AWS con SageMaker Studio

Índice

Esquema

Ideas clave

- 4.1. Introducción y objetivos
- 4.2. Características de la plataforma de desarrollo de modelos SageMaker Studio
- 4.3. Preparación de datos con SageMaker Data Wrangler
- 4.4. Ingeniería de características con SageMaker Feature Store
- 4.5. Diseño y entrenamiento de modelos
- 4.6. Sesgo y explicabilidad de modelos con SageMaker Clarify
- 4.7. Monitorización y mantenimiento de modelos con herramientas de SageMaker
- 4.8. Cuaderno de ejercicios
- 4.9. Referencias bibliográficas

A fondo

- Amazon SageMaker examples
- Amazon SageMaker documentation

Test

CREACIÓN DE MODELOS DE IA EN AWS CON SAGEMAKER STUDIO				
DATA WRANGLER	FEATURE STORE	EXPERIMENTS	CLARIFY	MODEL MONITOR
<ul style="list-style-type: none"> ▶ Importación de datos <ul style="list-style-type: none"> • Amazon S3 • Amazon Athena • Amazon Redshift • Snowflake • PC ▶ Exploración de datos <ul style="list-style-type: none"> • Histogramas • Scatter plots • Modelado rápido • Fuga de información • Detección de datos ausentes • Detección de outliers • Calidad de datos ▶ Transformación de datos <ul style="list-style-type: none"> • Eliminación de columnas • Eliminación de outliers • Imputación de datos ausentes • Codificación de variables categóricas • Generación de <i>n-grams</i> ▶ Ejecución de flow <ul style="list-style-type: none"> • Almacenamiento en S3 • Pipeline • Código fuente • Feature Store 	<ul style="list-style-type: none"> ▶ Componentes <ul style="list-style-type: none"> • Fuentes de datos • Definiciones de variables • Datos ingestados ▶ Beneficios <ul style="list-style-type: none"> • Independencia de variables y datos respecto a fuentes • Centralización de variables • Trazabilidad y auditoría ▶ Feature groups <ul style="list-style-type: none"> • <code>load_feature_definitions()</code> • <code>ingest()</code> • <code>put_record()</code> ▶ Acceso al Feature Store <ul style="list-style-type: none"> • Modo <i>offline</i> • Modo <i>online</i> 	<ul style="list-style-type: none"> ▶ Algoritmos propios de SageMaker <ul style="list-style-type: none"> • Imágenes • Estimadores ▶ Algoritmos diseñados por el usuario <ul style="list-style-type: none"> • TensorFlow • PyTorch • Hugging Face • MXNet • scikit-learn ▶ Control de versiones y cuadernos colaborativos <ul style="list-style-type: none"> • Repositorio Git • Compartición de notebooks ▶ Ejecución de experimentos <ul style="list-style-type: none"> • Creación de experimentos • Creación de trials • Acceso a resultados del experimento 	<ul style="list-style-type: none"> ▶ Detección de sesgos <ul style="list-style-type: none"> • Sesgo de datos • Sesgo de modelos ▶ Explicabilidad de modelos <ul style="list-style-type: none"> • Importancia de variables • SHAP • Valores Shapley 	<ul style="list-style-type: none"> ▶ Concept drift <ul style="list-style-type: none"> • <i>Ground truth</i> • Distribución estadística de los datos ▶ Monitor de calidad de datos <ul style="list-style-type: none"> • Creación de <i>job schedule</i> • Detección de incidentes en <i>Monitoring job history</i> ▶ Monitor de calidad de modelos <ul style="list-style-type: none"> • Creación de <i>job schedule</i> • Detección de incidentes en <i>Monitoring job history</i>

4.1. Introducción y objetivos

En este tema, se describen los componentes clave de la plataforma de inteligencia artificial de Amazon Web Services, **SageMaker**:

En primer lugar, se presenta la herramienta **Data Wrangler**, que permite llevar a cabo las actividades de ingestión, preparación y transformación de datos a través de un diagrama de bloques, sin necesidad de programación.

A continuación, se introduce la herramienta **Feature Store**, que proporciona un almacén de características para centralizar la gestión de las características procedentes de múltiples fuentes de datos.

También se describe la herramienta **Processing**, que permite encapsular el proceso de entrenamiento en un contenedor. Este contenedor puede ejecutarse en la infraestructura de computación gestionada por Amazon, de forma transparente al usuario.

Veremos que es posible utilizar **algoritmos** disponibles en SageMaker o diseñar uno propio a partir de los *frameworks* más comunes, así como compartir los **cuadernos Jupyter** generados en el proceso para ampliar la base de conocimiento o facilitar la creación colaborativa de modelos.

Asimismo, se presentan las herramientas **Clarify** y **Model Monitor**. La primera se utiliza para detección de sesgos y medida de importancia de características, mientras que la segunda se utiliza para detectar la degradación del rendimiento de un modelo desplegado en producción.

Por último, este tema se plantea los siguientes objetivos para el alumno:

- ▶ Ser capaces de utilizar AWS SageMaker para diseñar, construir y desplegar un modelo de inteligencia artificial.
- ▶ Adquirir el conocimiento necesario para explotar las ventajas de un entorno de computación en la nube a la hora de preparar y transformar datos.
- ▶ Explotar las ventajas de un Feature Store para eliminar duplicidad de esfuerzos y facilitar la colaboración entre equipos.
- ▶ Promover el uso de una IA responsable, que sea capaz de detectar sesgos en los datos y en el modelo entrenado, así como explicar los motivos que conducen al valor de una predicción.
- ▶ Explotar las ventajas de la monitorización de modelos para ser capaces de entregar a un cliente el ciclo completo de gestión de modelos, incluyendo las actividades que suceden tras su puesta en producción.

4.2. Características de la plataforma de desarrollo de modelos SageMaker Studio

Herramientas disponibles

En esta sección se presentan las herramientas que proporciona SageMaker para cada una de las fases del proceso de creación de modelos.

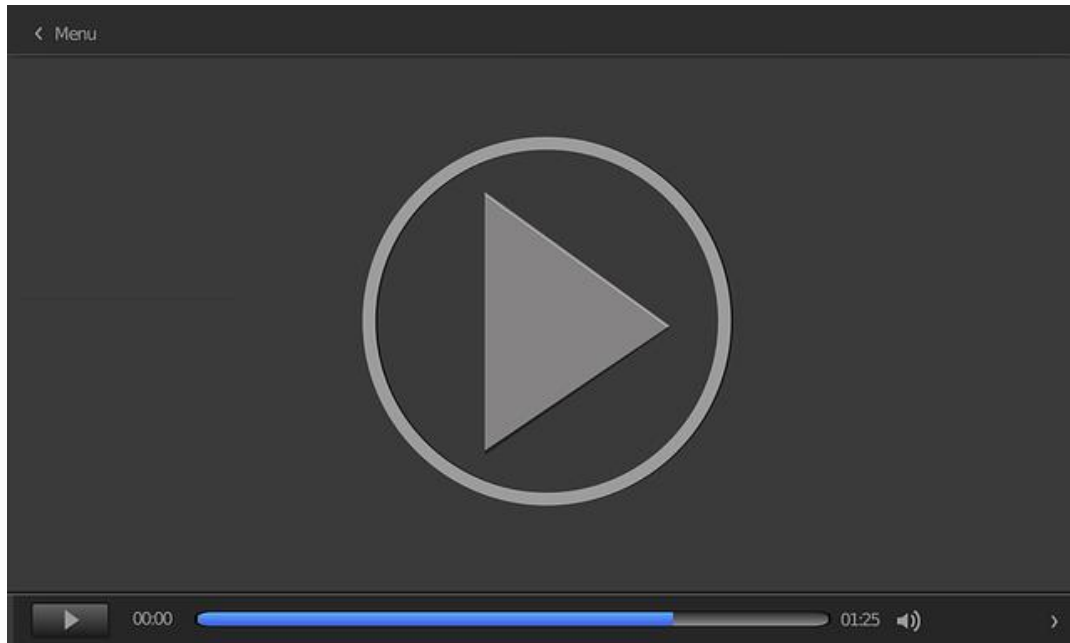
Preparación de datos

Las herramientas de preparación de datos son las siguientes:

Nombre	Descripción
Data Wrangler	Herramienta de ingeniería de datos basada en interfaz web, que se emplea para la ingesta de datos desde otras ubicaciones en la nube (por ejemplo, Amazon S3, Databricks o Snowflake), así como para llevar a cabo labores de preparación de datos. Permite construir diagramas con bloques para configurar cada tarea del proceso de preparación de datos. Estos bloques puedan ser reutilizados en otros diagramas. Dispone de una librería de 300 funciones predefinidas para construir bloques. También se pueden crear bloques personalizados con código propio en Python, Spark, y SQL.
Clarify	Esta herramienta detecta sesgos (<i>bias</i>) o infrarrepresentación de grupos o categorías en los datos, que conducen a imprecisión o errores en las predicciones. También implementa mecanismos de detección de importancia de características, lo que a su vez facilita la detección de sesgos.
Processing	Esta herramienta permite encapsular código fuente (<i>scripts</i>) en contenedores Docker que satisfacen las dependencias de <i>software</i> y elegir la infraestructura de computación en la que son ejecutados. Processing es utilizado en las propias herramientas de SageMaker. Por ejemplo, para la ejecución de los diagramas de Data Wrangler.
Feature Store	Es un almacén gestionado de características. Permite crear, almacenar, modificar y compartir características construidas a partir de <i>datasets</i> . Proporciona versionado de características y la funcionalidad de <i>time travel</i> , para obtener el valor de una característica en un instante anterior. Ofrece la modalidad <i>offline</i> para utilizar las características del <i>store</i> en procesos de entrenamiento o inferencia <i>batch</i> , y la modalidad <i>online</i> para procesos de inferencia en tiempo real.

Tabla 1. Herramientas de preparación de datos en SageMaker. Fuente: elaboración propia.

En el siguiente vídeo, *Uso de SageMaker Processing para encapsular fases del proceso de creación de modelos*, se muestra un ejemplo real de uso de la herramienta Processing de SageMaker.



Accede al vídeo:

<https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=178718f9-a01a-4466-961b-b14c00e27816>

Entrenamiento de modelos

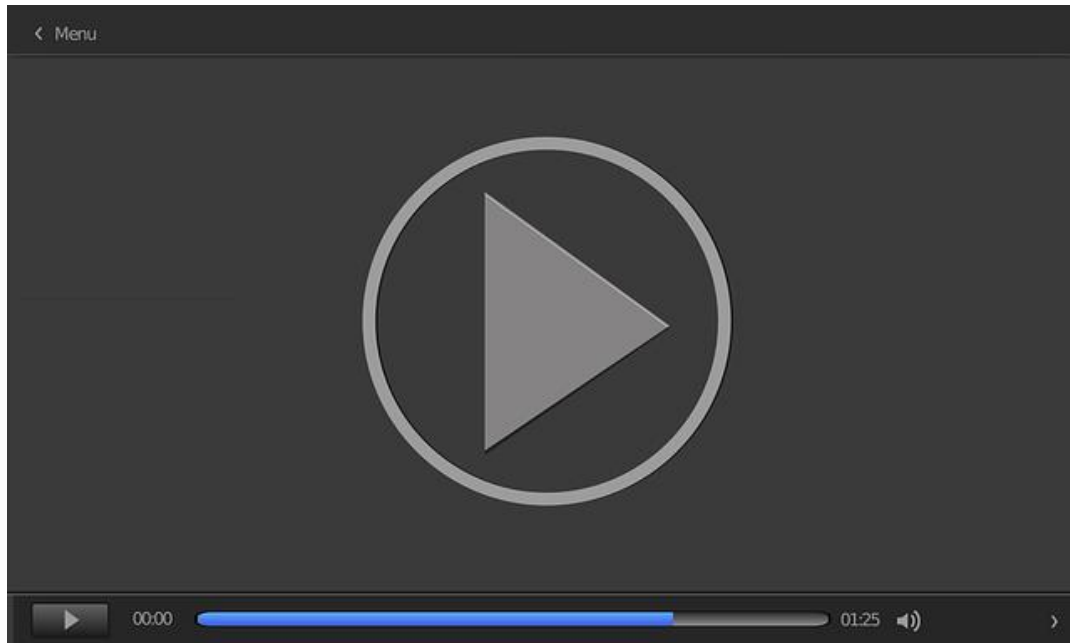
Las herramientas para el entrenamiento de modelos son las siguientes:

Nombre	Descripción
Autopilot	Herramienta de AutoML que prepara datos, entrena modelos y ajusta parámetros de forma automática. Se configura a través de la interfaz web de SageMaker Studio. Solo es preciso indicar la ubicación del <i>dataset</i> y la característica objetivo que se quiere predecir. Proporciona visibilidad completa de cómo se ha realizado el proceso, entregando los <i>notebooks</i> Jupyter utilizados. Estos <i>notebooks</i> pueden emplearse para mejorar la implementación de AutoPilot. Permite comparar el rendimiento obtenido con cada configuración de hiperparámetros, seleccionar el modelo con mejores resultados y desplegarlo en un <i>endpoint</i> (URL).
JumpStart	Proporciona un catálogo de soluciones y modelos que resuelven los casos de uso más habituales de cada sector, incluyendo visión artificial y procesamiento de lenguaje. Tanto los modelos como las soluciones pueden desplegarse directamente desde la web. También es posible reentrenar los modelos con datos propios para mejorar su rendimiento, sin necesidad de código fuente.
Studio Notebooks	Esta herramienta está basada en JupyterLab. Ofrece el servicio de <i>notebooks</i> Jupyter de forma gestionada. Permite escalar de forma dinámica la infraestructura de computación utilizada por los <i>notebooks</i> .
Algoritmos integrados	Colección de implementaciones de los algoritmos de aprendizaje supervisado y no supervisado más comunes. Para utilizarlos, solo es preciso especificar hiperparámetros y datos. La provisión de la infraestructura para el entrenamiento se realiza de forma automática. Las implementaciones de estos algoritmos soportan de forma nativa arquitecturas distribuidas.

Nombre	Descripción
Entrenamiento gestionado	<p>SageMaker proporciona la capacidad de provisionar bajo demanda la infraestructura de computación necesaria para la ejecución de procesos de entrenamiento. El usuario solo debe especificar las características del <i>hardware</i> y crear la rutina (<i>job</i>) de entrenamiento.</p> <p>SageMaker se encarga de provisionar el <i>hardware</i>, ejecutar el código fuente, proporcionar las salidas y, finalmente, decomisionar el <i>hardware</i>.</p>
Entrenamiento distribuido	<p>SageMaker proporciona librerías de entrenamiento distribuido basadas en paralelismo de datos y paralelismo de modelos.</p> <ul style="list-style-type: none"> ▶ Con el paralelismo de datos, cada nodo de la arquitectura distribuida recibe una parte de los datos y la utiliza para entrenar el algoritmo. ▶ Con el paralelismo de modelos, cada nodo implementa una parte del algoritmo y la utiliza para procesar todos los datos de entrenamiento. <p>El uso de estas librerías solo requiere cambios menores en el código fuente.</p>
SageMaker Debugger	<p>SageMaker permite depurar fácilmente un proceso de entrenamiento. Proporciona métricas relacionadas con el proceso y con la infraestructura de computación utilizada. De esta forma, es posible detectar de forma temprana errores o incidentes. La información se proporciona a través de un cuadro de mando.</p>
SageMaker Experiments	<p>Herramienta de gestión de pruebas y configuraciones (experimentos) en el proceso de creación de modelos. Permite registrar todos los detalles de cada prueba, incluyendo: datos empleados, transformaciones aplicadas, valores de hiperparámetros, estimador elegido, métricas de rendimiento, y gráficas generadas. Estas trazas permiten comparar los resultados de diferentes experimentos y seleccionar la configuración óptima para despliegue.</p>

Tabla 2. Herramientas de entrenamiento de modelos en SageMaker. Fuente: elaboración propia.

En el siguiente vídeo, *Uso de AutoPilot en SageMaker para automatizar ingeniería de características, creación y despliegue de modelos*, se muestra un ejemplo real de uso de la herramienta AutoPilot de SageMaker.



Accede al vídeo:

<https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=62417dff-5e84-43af-842f-b149010312f6>

Despliegue de modelos

Las herramientas para el despliegue de modelos son las siguientes:

Nombre	Descripción
Despliegue gestionado	<p>Permite desplegar modelos de forma automática en infraestructura gestionada por AWS tanto para procesamiento <i>batch</i> como para inferencia en tiempo real.</p> <p>El <i>endpoint</i> desplegado devuelve predicciones a través de una URL, con baja latencia, y puede escalar automáticamente cuando se incrementa el volumen de peticiones. También es posible desplegar varias variantes de un modelo bajo un mismo <i>endpoint</i> para realizar <i>A/B testing</i>. La carga puede repartirse conforme a los porcentajes especificados y actualizarse de forma dinámica para dar prioridad a la variante que ofrece mejores datos de rendimiento en producción.</p>
Model Monitor	<p>Construye un <i>baseline</i> de los datos de entrada en producción e identifica desviaciones (<i>concept drift</i>) respecto a dicho <i>baseline</i>.</p> <p>La detección temprana de <i>concept drift</i> permite llevar a cabo un reentrenamiento para garantizar que se mantiene la precisión de las predicciones a lo largo del tiempo. El usuario puede especificar sus propias métricas para evaluar el <i>concept drift</i>.</p>

Tabla 3. Herramientas de despliegue de modelos en SageMaker. Fuente: elaboración propia.

Operacionalización de modelos

Las herramientas de MLOps disponibles son las siguientes.

Nombre	Descripción
Pipelines	Es un sistema de orquestación que permite definir un <i>workflow</i> con cada una de las fases del ciclo de vida de creación de modelos, desde el procesamiento de datos hasta el despliegue en <i>endpoints</i> . Estos <i>workflows</i> pueden ejecutarse bajo demanda, de forma programada o cuando se producen determinados eventos, que actúan como <i>triggers</i> o disparadores.
Projects	Permite implementar prácticas de CI/CD (integración y despliegue continuos) en el proceso de creación de modelos. Se basa en plantillas. En estas es posible configurar: los repositorios de código fuente, los <i>pipelines</i> de entrenamiento, los controles de calidad del modelo y la automatización del despliegue.
Model Registry	Permite registrar el modelo generado a partir de un proceso de entrenamiento en un repositorio. El registro facilita la auditoría y trazabilidad de los modelos, y permite su despliegue automatizado en infraestructura gestionada por AWS.

Tabla 4. Herramientas de MLOps en SageMaker. Fuente: elaboración propia.

Puesta en marcha del entorno

Para empezar a utilizar SageMaker, es necesario configurar previamente un **dominio**. Este incluye las siguientes configuraciones:

- ▶ **Método de autenticación:** permite configurar la forma en que los usuarios acceden a la plataforma. Las opciones disponibles son: *single sign-on* (SSO) o AWS Identity and Access Management.
- ▶ **Rol de ejecución por defecto:** especifica de qué permisos se dispone para realizar acciones bajo SageMaker Studio. Los usuarios creados en el dominio pueden heredar este rol o ser asignados a otro.
- ▶ **Opciones de compartición de *notebooks*:** permite trabajar de forma colaborativa, proporcionando acceso compartido a los *notebooks* y sus salidas a los usuarios del dominio.
- ▶ **Disponibilidad de JumpStart y SageMaker Projects:** Jumpstart es un repositorio de modelos preentrenados para los casos de uso más comunes, mientras que SageMaker Projects se emplea para automatizar y orquestar todas las etapas del ciclo de vida de creación de modelos a través de *pipelines*.
- ▶ **Redes y almacenamiento:** permite incrementar el nivel de seguridad y personalización de las redes utilizadas por SageMaker y de los sistemas de almacenamiento empleados para los datos.
- ▶ **Etiquetas (*tags*):** el uso de *tags* facilita la identificación del propósito y características de cada dominio creado en SageMaker.

Uso del SDK de Python

Todas las herramientas de SageMaker pueden ser utilizadas desde la interfaz web o desde el SDK de Python, es decir, a través de **código fuente**. También puede emplearse el cliente de línea de comandos de AWS.

Por otro lado, para acceder desde código fuente a otros servicios y recursos disponibles en Amazon (fuera de SageMaker), se utiliza el SDK de AWS (Boto3). Por ejemplo, **Boto3** se utiliza para acceder a la plataforma de almacenamiento de AWS (S3).

El **código de inicialización** necesario para empezar a utilizar el SDK es el siguiente:

```
import sagemaker

session = sagemaker.Session()

bucket = session.default_bucket()

role = sagemaker.get_execution_role()
```

En el código anterior se ejecutan los siguientes pasos:

- ▶ Se importa la librería de funciones de SageMaker .
- ▶ Se crea una sesión de SageMaker , a través de la cual se utilizan todas las funcionalidades que ofrece el SDK.
- ▶ Se asigna un puntero al bucket S3 por defecto de la sesión de SageMaker , en la que podemos almacenar artefactos y ficheros generados en el proceso de creación de modelos.
- ▶ Se asigna un puntero al rol de ejecución del que dispone el usuario que ejecuta el *notebook*. Este rol es un parámetro de muchas funciones disponibles en el SDK, y determina el alcance de las acciones que pueden llevarse a cabo en la sesión de SageMaker .

4.3. Preparación de datos con SageMaker Data Wrangler

Acceso a Data Wrangler

Desde SageMaker Studio, es posible crear un nuevo **diagrama de bloques (flow)** haciendo clic en «File», «New», «Data Wrangler Flow», en el menú ubicado en la parte superior de la interfaz web. El nombre asignado por defecto es `untitled.flow`, pero puede modificarse para que refleje el propósito del diagrama.

Cuando el usuario crea un nuevo *flow*, SageMaker provisiona automáticamente una nueva instancia de *hardware* de tipo `ml.m5.4xlarge`. Esta instancia se utiliza exclusivamente para el *flow*.

Al acceder al *flow*, se presenta al usuario un lienzo o canvas vacío, en el que podemos ir incorporando los diferentes bloques (o *steps*) que componen el diagrama.

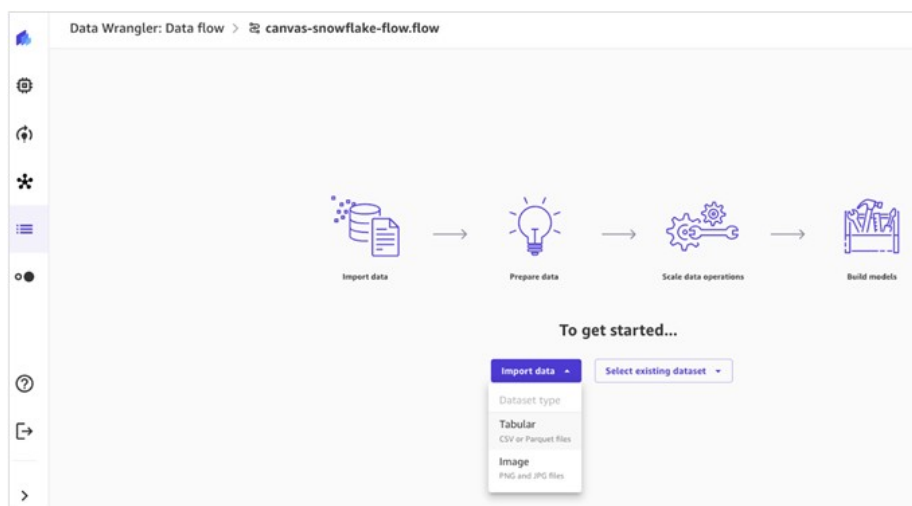


Figura 1. Canvas inicial de Data Wrangler. Fuente: Ma, Govindaram y Nguyen, 2023.

Importación de datos

Data Wrangler permite importar datos de las siguientes **fuentes**:

- ▶ **Amazon S3**: sistema de almacenamiento basado en objetos de AWS. Para importar datos, solo es necesario especificar el *bucket* y la ruta de acceso al fichero (por ejemplo, CSV).
- ▶ **Amazon Athena**: entorno analítico basado en SQL. Para importar datos, es preciso indicar el catálogo de datos, la base de datos y una consulta SQL que accede a las tablas de la base de datos.
- ▶ **Amazon Redshift**: *data warehouse* de AWS.
- ▶ **Snowflake**: servicio externo de almacenamiento, procesamiento y analítica de datos.
- ▶ **PC del usuario**: es posible seleccionar un fichero del sistema de ficheros local de nuestro PC y subirlo a Data Wrangler.

Una vez cargados los datos, el siguiente paso es verificar y corregir los tipos de datos asignados a cada columna del *dataset* resultante. Para ello, debemos hacer clic en el símbolo «+» del *dataset*, y a continuación en «Edit data types».

Los tipos de datos disponibles son: entero largo (*long*), decimal (*float*), booleano (*boolean*), cadena de caracteres (*string*) y fecha (*timestamp*). La asignación correcta de tipos de datos es clave para poder aplicar, posteriormente, las transformaciones que sean necesarias. Por ejemplo, para poder reemplazar datos ausentes por la media de esa columna en el conjunto de datos, la columna debe ser de tipo numérico.

En caso de que la información que queremos procesar no esté disponible en un único fichero, sino en varios, es posible importar cada uno de ellos y realizar un *join* para combinar los datos en un único *dataset*. Para llevar a cabo esta operación, es necesario que los datos de cada fichero tengan una columna en común.

Exploración de datos

Histogramas

Uno de los primeros pasos en el proceso de exploración de datos es consultar los histogramas de cada característica. Para ello, debemos hacer clic en el símbolo «+» del *dataset*, y a continuación seleccionar «Add analysis». En «Analysis type» elegimos «Histogram», y especificamos la característica para la cual queremos extraer la distribución de frecuencia.

También podemos visualizar histogramas para cada valor de la característica que queremos predecir. Para ello, seleccionamos esta característica en «Facet by». Esto nos permite identificar la potencial capacidad predictiva de la característica.

Scatter plots

Otro tipo de análisis frecuente es el que permite enfrentar las observaciones de dos características en un *scatter plot*. Para ello, se selecciona «Scatter Plot» en «Analysis type», y se especifican las dos características en el eje x e y.

La gráfica que muestra Data Wrangler puede ayudar a detectar correlación entre estas características. Si se trata de dos características de entrada, la existencia de esta correlación implica **multicolinealidad**, que es un efecto indeseable. Si se trata de una característica de entrada y la característica que queremos predecir, entonces implica **potencia predictiva**. Por tanto, en este segundo caso habremos identificado una característica útil para nuestro futuro modelo.

Modelado rápido

Data Wrangler permite construir rápidamente un modelo a partir de un fragmento de los datos para anticipar la viabilidad de un modelo completo. Para ello, podemos seleccionar «Quick Model» en «Analysis type», e indicar la característica que queremos predecir, denominada «**característica objetivo**».

El algoritmo utilizado para este ejercicio es **random forest**. En caso de que la característica objetivo sea de tipo categórico, se construye un modelo de clasificación y se calcula el *F1 score*, que mide la calidad de este.

Data Wrangler también presenta un **diagrama de importancia** de características para identificar cuáles han tenido mayor capacidad predictiva en este ejercicio de modelado. Esta información es útil para las labores posteriores de preparación de datos.

Fuga de información

Data Wrangler permite detectar escenarios en los que el modelo tiene acceso a datos durante el entrenamiento que no estarían disponibles en el momento de hacer predicciones en un entorno real, lo que conduce a un rendimiento engañosamente alto durante la fase de prueba y a un rendimiento pobre en producción. Para identificar este tipo de situaciones, debemos seleccionar «Target Leakage» en «Analysis type».

Otros tipos de análisis

Data Wrangler permite llevar a cabo múltiples análisis adicionales, incluyendo detección de datos ausentes (*missing data*), detección de *outliers* (observaciones anómalas) o calidad de datos.

Transformación de datos

Una vez analizados los datos, pueden aplicarse en Data Wrangler las transformaciones necesarias para corregir las **deficiencias identificadas** o **preparar los datos** de manera óptima para el proceso de creación del modelo. Para ello, debemos hacer clic en el símbolo «+» del *dataset* y seleccionar «Add transform». Cada transformación representa un *step* en el diagrama de bloques.

SageMaker proporciona una colección de funciones predefinidas para las tareas más habituales. También es posible utilizar código fuente para definir transformaciones personalizadas.

Algunas de las funciones más comunes son:

- ▶ **Eliminación de columnas:** por ejemplo, aquellas que presentan multicolinealidad con otras características de entrada o aquellas que no tienen capacidad predictiva o contienen múltiples errores.
- ▶ **Eliminación de *outliers*:** permite eliminar las observaciones que pueden provocar sobreajuste en el proceso de construcción del modelo.
- ▶ **Tratamiento (*imputing*) de datos ausentes:** permite asignar valores a los datos ausentes del *dataset*. Funciona con características numéricas o categóricas. En el primer caso, puede asignarse la media o la mediana de la característica, o bien un valor fijo; en el segundo, puede utilizarse un valor por defecto o la moda (valor más frecuente) de esta característica.
- ▶ **Codificación de características categóricas:** permite transformar características categóricas en características numéricas, usando diferentes técnicas de codificación, como *label encoding* o *one-hot encoding*.
- ▶ **Generación de *n-grams*:** permite transformar un texto en combinaciones de palabras indexadas para facilitar su tratamiento posterior.

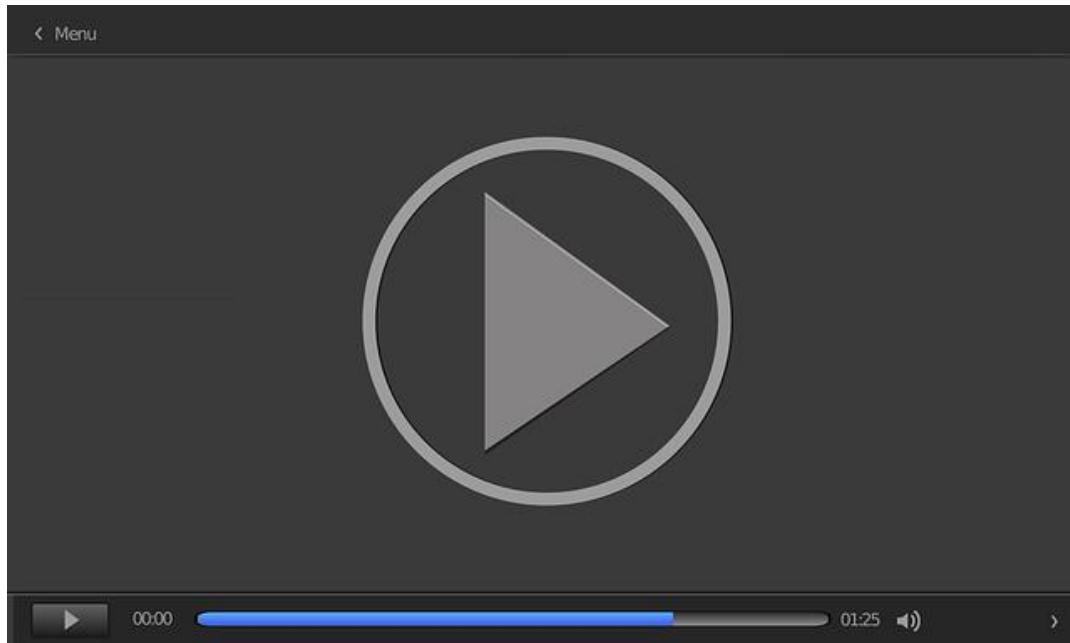
Ejecución del flow

Las transformaciones de datos definidas en el punto anterior no son aplicadas hasta que el *flow* es ejecutado.

La salida del *flow* puede ser alguna de las siguientes:

- ▶ **Almacenamiento en S3:** genera un fichero de salida que contiene los datos transformados.
- ▶ **Pipeline:** el diagrama se incorpora a un *pipeline*. De esta forma, se convierte en un proceso repetible que se activa mediante algún *trigger* o de forma planificada en el flujo de creación de modelos.
- ▶ **Código fuente:** proporciona el código fuente en Pyspark, asociado al conjunto de transformaciones incluidas en el diagrama. Este código fuente puede ejecutarse posteriormente en un entorno Spark.
- ▶ **Feature store:** proporciona un *notebook* Jupyter que contiene el código necesario para añadir los datos transformados como un grupo de características al Feature Store de SageMaker.

En el siguiente vídeo, *Uso de Data Wrangler para análisis exploratorio y transformaciones de datos*, se muestra un ejemplo real de uso de Data Wrangler en SageMaker.



Accede al vídeo:

<https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=e88e32f3-a413-4b8c-973f-b14d00ca63dd>

4.4. Ingeniería de características con SageMaker Feature Store

Conceptos generales

Un *feature store* permite **almacenar características y datos** en un repositorio único, alimentado a partir de múltiples fuentes de datos independientes. Cada característica puede estar vinculada de forma unívoca con una característica del *dataset* original, cuyos datos están almacenados en una fuente de datos utilizada por el *feature store*, o ser construidas a partir de una o más características de los *datasets* originales, utilizando fórmulas o técnicas de transformación como las descritas en el apartado anterior. Por ejemplo, una característica del *feature store* puede ser el promedio de tres características de un *dataset* almacenado en una fuente de datos.

Un almacén de características consta de tres **componentes**:

- ▶ Una colección de **fuentes de datos** de las que extrae características y datos.
- ▶ Una colección de **definiciones de características**.
- ▶ **Datos ingestados** de las fuentes y asociados a las características definidas.

Cada característica está sujeta al **versionado**, y su definición puede actualizarse para reflejar cambios en los datos originales o evoluciones en el procesamiento de esta. En cierto modo, un *feature store* es un ejercicio de abstracción de características, que intercala una capa adicional entre los datos originales y el usuario que los utiliza.

Los **beneficios** obtenidos por el uso de un *feature store* son las siguientes:

- ▶ **Independencia de características y datos respecto a las fuentes de origen.** El *feature store* proporciona libertad a los científicos de datos para extraer o fabricar las características que necesitan, al margen del equipo de ingeniería de datos que mantiene las fuentes de datos.
- ▶ **Centralización de características en un único repositorio.** Este enfoque evita que varios científicos de datos construyan las mismas características por separado, lo que implica duplicidad de esfuerzos y falta de control. Además, permite que se utilicen exactamente las mismas características en el proceso de entrenamiento y en el momento de realizar inferencias.
- ▶ **Trazabilidad y auditoría.** Con un *feature store* es posible explicar qué datos se han utilizado para entrenar un modelo y cómo se han generado las características utilizadas en el proceso.

Un *feature store* puede funcionar en **modo online u offline**. El primero es utilizado por los servicios de inferencia para incorporar las características que necesitan antes de entregar una predicción. El requisito principal de este modo de funcionamiento es la baja latencia. El segundo se utiliza en los procesos de entrenamiento que requieren un mayor volumen de datos y, sin embargo, admiten una latencia mayor.

Creación de feature groups en SageMaker

El primer paso para utilizar un *feature store* en SageMaker es crear un *feature group*, que agrupa las definiciones de un conjunto de características relacionadas (por ejemplo, aquellas que proceden de un mismo *dataset*). Para ello, seleccionamos «Feature Store» dentro de la sección «SageMaker Resources», y a continuación hacemos clic en «Create feature group».

El siguiente paso es especificar, en formato JSON o a través una tabla editable, el **nombre y tipo de las características** que componen el *feature group*. Si ejecutamos este paso desde el SDK, podemos llamar a la función `load_feature_definitions()` con un *dataframe* como parámetro. Esta función extrae automáticamente las definiciones de las características del *feature group* a partir de las características del *dataframe*.

A continuación, es preciso seleccionar cuál de las características actuará como **identificador del registro**. En terminología de base de datos, esta característica sería la clave primaria del *feature group*.

El siguiente paso es la **ingestión de los datos**, lo que permite proporcionar contenido a las características del *feature group*. Este paso se realiza con la función `ingest()` del SDK para datos *batch*, o con `put_record()` para datos que proceden de una fuente de *streaming*. Si el *feature store* funciona en *offline* y *online* simultáneamente, los datos ingestados estarán disponibles en ambas modalidades.

Adicionalmente, la creación de un *feature group* puede automatizarse desde Data Wrangler. Para ello, es necesario incorporar un Export step, seleccionar «Feature Store» en la lista de opciones de exportación y ejecutar el cuaderno Jupyter resultante en SageMaker Processing.

Acceso a características del feature store

Desde la interfaz web de SageMaker, es posible acceder a los *feature groups* creados en el *feature store* y consultar los detalles de cada uno de ellos, incluyendo las definiciones de características. Para acceder a su contenido, se puede utilizar la **SDK** en Python. El acceso es diferente para el modo *offline* y *online*.

Modo *offline*

El almacenamiento de los datos en el modo *offline* se realiza en un **bucket de S3**. Se utiliza AWS Glue Catalog para generar un catálogo de datos asociado, que a su vez se expone a través de una API SQL en Amazon Athena. Podemos utilizar esta interfaz **SQL** para extraer datos de un *feature group* determinado. En el siguiente ejemplo, se muestra cómo hacerlo con el SDK de Python:

```
consulta_athena = feature_group.athena_query()

nombre_tabla = consulta_athena.table_name

query_sql_text = ('SELECT * FROM "%s"' % nombre_tabla)

query_sql.run(query_string=query_sql_text,
output_location=f's3://{bucket}/{prefix}/query_results/')

query_sql.wait()

dataframe = query.as_DataFrame()
```

Los datos devueltos incluyen tres características adicionales, añadidas para facilitar el mantenimiento del *offline feature store*: *api_invocation_time* (*timestamp* de la petición de escritura), *write_time* (*timestamp* de escritura del dato) y *is_deleted* (*flag* que indica si se ha solicitado el borrado del registro).

Para evitar **target leakage**, es posible utilizar la funcionalidad de *point-in-time*. Esta funcionalidad permite limitar las características que se devuelven en una consulta a aquellas que estaban disponibles en un instante temporal dado, y ninguna de las que se añadió posteriormente. Por ejemplo, si queremos detectar fraude, no podemos utilizar la característica que representa la acción tomada como resultado de un fraude (que se añade posteriormente), porque esa característica no estará disponible en el momento de realizar predicciones con datos nuevos.

Modo *online*

Para acceder a registro en modo *online*, puede utilizarse el SDK de Python, como se indica a continuación:

```
identificador = 'valor'

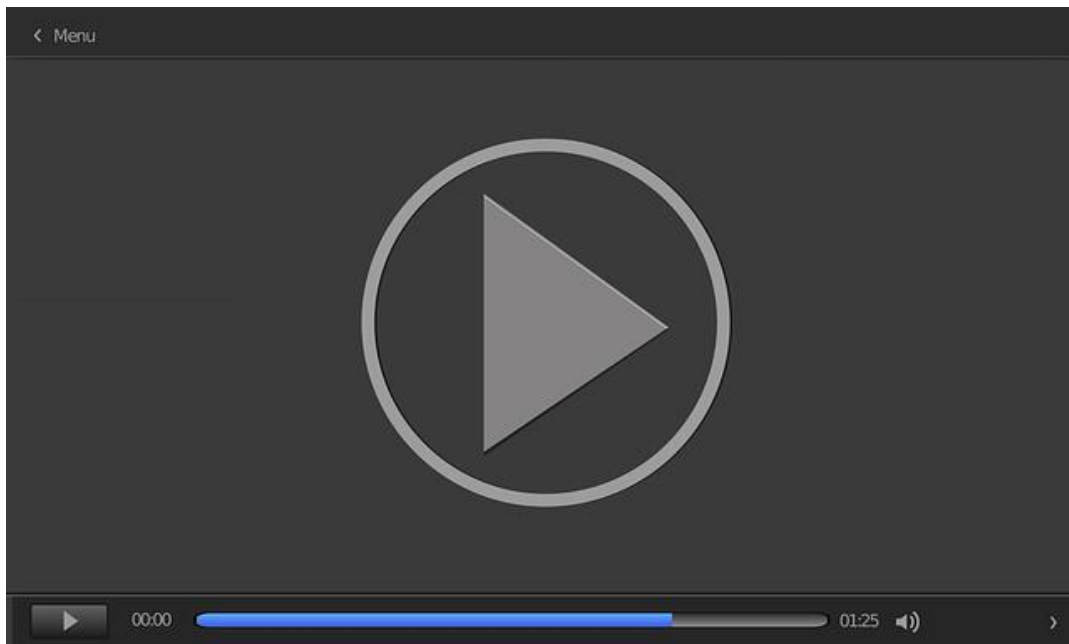
featurestore_runtime = sess.boto_session.client(service_name='sagemaker-
featurestore-runtime', region_name=region)

resultado =
featurestore_runtime.get_record(FeatureGroupName=nombre_feature_group,
RecordIdentifierValueAsString=identificador)

resultado['Record']
```

En el ejemplo anterior, se inicializa un valor de la característica seleccionada como clave primaria del *feature group*. A continuación, se instancia el *runtime* del *feature store* y se llama a `get_record()`, especificando el *feature group* y el valor de clave primaria. Por último, se accede al contenido del registro con la clave `Record` del resultado.

En el siguiente vídeo, *Uso de Feature Store en AWS Sage Maker para gestionar de forma unificada colecciones de características y transformaciones de datos*, se muestra un ejemplo real de configuración de un Feature Store en SageMaker:



Accede al vídeo:

<https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=998f6da5-38c2-4465-bb17-b14c00f8f818>

4.5. Diseño y entrenamiento de modelos

Uso de algoritmos propios de SageMaker

Los algoritmos propios de SageMaker permiten construir modelos, especificando únicamente los datos, el nombre del algoritmo elegido, los valores de hiperparámetros y la infraestructura de computación. De esta forma, evitamos tener que escribir el código asociado al algoritmo en un *script*. Además, las implementaciones de AWS están diseñadas para ejecutarse en arquitecturas distribuidas o en GPUs para reducir al máximo el tiempo de entrenamiento.

En la Tabla 5 se resumen los **tipos de algoritmos** disponibles:

Categoría	Algoritmos
Aprendizaje supervisado	<i>XGBoost</i>
Aprendizaje no supervisado	<i>k-means</i>
Análisis de imágenes	Clasificación de imágenes, detección de objetos, segmentación semántica.
Análisis de texto	<i>word2vec</i> , clasificación de textos, <i>sequence-to-sequence</i>

Tabla 5. Algoritmos propios de SageMaker. Fuente: elaboración propia.

El **proceso de entrenamiento** con un algoritmo de SageMaker se ejecuta dentro de un contenedor, instanciado en una infraestructura independiente de la utilizada para correr el *notebook* Jupyter desde el que se lanza. La imagen del contenedor incluye todas las dependencias de *software* para ejecutar el algoritmo. Las imágenes están almacenadas en un registro denominado «Amazon Elastic Container Registry (ECR)».

En el siguiente fragmento de código se utiliza el SDK de Python para seleccionar una imagen y definir el contenedor que ejecuta el proceso de entrenamiento con el algoritmo incluido en la imagen:

```
image=sagemaker.image_uris.retrieve(

    framework=' xgboost ',

    region=region,

    version='1')

estimator = sagemaker.estimator.estimator(

    image,

    role,

    instance_count=1,

    instance_type='ml.c5.2xlarge',

    volume_size=30,

    max_run=3600,

    input_mode='File',

    enable_sagemaker_metrics=True,

    output_path=s3_path,

    hyperparameters={ ... })
```

En el ejemplo anterior, se selecciona la imagen con la versión 1 del algoritmo `xgboost` (`retrieve()`). A continuación, se define un contenedor a partir de la imagen (`estimator`). Este contenedor se ejecuta en una única máquina virtual (`instance_count`) con el perfil de *hardware* `ml.c5.2xlarge` (`instance_type`). Se indica que estará en ejecución como máximo una hora (`max_run`), que se registrarán las trazas de actividad (`enable_sagemaker_metrics`) y que se almacenarán los resultados en S3 (`output_path`). También se indica el valor de los hiperparámetros (`hyperparameters`).

El proceso de entrenamiento arranca cuando se llama a la función `fit()` :

```
data = {'train': s3_path_train, 'validation': s3_path_validation }

estimator.fit(inputs=data, job_name='job', logs=True)
```

Las trazas de actividad generadas durante el proceso pueden consultarse desde Amazon CloudWatch Metrics, y el modelo generado está disponible a través de la característica `estimator.model_data` .

Uso de algoritmos diseñados por el usuario

El usuario también puede utilizar **código propio** para ejecutar un algoritmo o realizar un diseño personalizado de una red neuronal. En ese caso, SageMaker facilita la labor, proporcionando contenedores que incluyen alguno de los principales *frameworks* de algoritmos: TensorFlow, PyTorch, Hugging Face, MXNet y scikit-learn.

En el siguiente ejemplo, se muestra cómo utilizar TensorFlow desde SageMaker , con un contenedor facilitado por AWS y código propio almacenado un `script` :

```
hyperparameters = {'epochs': 100, 'batch_size': 128, 'learning_rate': 0.05
, 'drop_out_rate': 0.1 }

estimator = TensorFlow(source_dir='code',

    entry_point='script.py',

    model_dir='./model',

    code_location='./code',

    instance_type='ml.p3.2xlarge',

    instance_count=1,

    enable_sagemaker_metrics=True,
```

```
hyperparameters=hyperparameters,

role=role,

framework_version='2.1',

py_version='py3')
```

Uso de control de versiones y notebooks colaborativos

Los *notebooks* Jupyter utilizados para la creación de modelos son artefactos incluidos en el repositorio Git, que implementa el control de versiones en SageMaker Studio. Por tanto, es posible consultar los **cambios incorporados** de una versión a otra a través de la interfaz web. Esta funcionalidad incrementa la **trazabilidad** de los modelos generados a partir de un proceso de entrenamiento incluido en un *notebook*.

También es posible compartir los *notebooks* disponibles con otros científicos de datos para que puedan aprovechar el trabajo previamente realizado por uno de ellos o analizar sus resultados. SageMaker proporciona la opción de **compartir el código** del *notebook* en modo de solo lectura, y que el usuario que lo consulta copie el contenido en un nuevo *notebook* para continuar el trabajo.

Ejecución de experimentos

Un experimento en SageMaker es una **colección de pruebas** (*trials*) de procesamiento de datos o entrenamiento cuyos resultados quedan registrados. El uso de estos garantiza la trazabilidad de cada una de las pruebas, de manera que podamos comparar los resultados obtenidos en cada una de ellas y seleccionar la que conduce al modelo óptimo.

En el siguiente fragmento de código se crea un nuevo experimento y sucesivas pruebas (*trials*) de entrenamiento con la SDK de Python:

```
experiment = Experiment.create(experiment_name='nombre',
description='descripción')
```



```
for valor_hiperparametro in [1, 2, 3]:

    prueba = Trial.create(experiment_name='nombre_experimento',
trial_name='nombre_prueba')

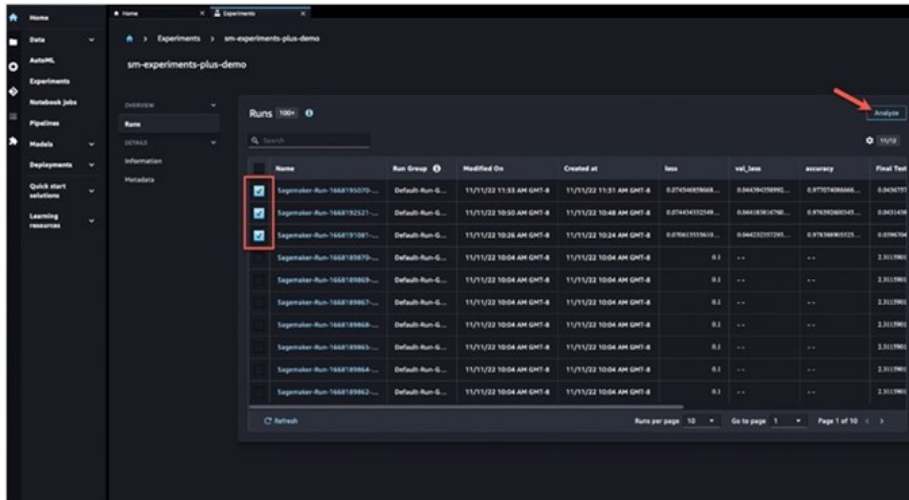
    experiment_config={ 'ExperimentName': 'nombre_experimento',

'TrialName': prueba.trial_name, 'TrialComponentDisplayName':
'Entrenamiento' }

    [ definir estimador con el valor actual del hiperparámetro ]
    estimator.fit(inputs=data, job_name='nombre_job',
experiment_config=experiment_config, wait=False)
```

Se define un nuevo estimador para cada valor de hiperparámetro que desea probarse en el experimento. La llamada a `fit()` desencadena un `job` de entrenamiento en cada iteración que recibe la configuración del experimento.

En la interfaz web es posible acceder a los **resultados del experimento**. Para ello, se debe seleccionar «Experiments and trials» en «SageMaker Resources». Se muestran cada uno de los *trials* con el valor de las métricas de rendimiento obtenidas para cada valor del hiperparámetro. También es posible visualizar gráficas con la evolución del rendimiento a lo largo del conjunto de valores de hiperparámetros que se han probado. En la Figura 2 se muestra un ejemplo de experimento:



The screenshot shows the AWS SageMaker Experiments console. The left sidebar contains navigation options like Home, Data, AutoML, Experiments, Notebook Jobs, Pipelines, Models, Deployments, Quick start solutions, and Learning resources. The main area displays the 'Runs' tab for an experiment named 'sm-experiments-plus-demo'. A table lists the runs with columns: Name, Run Group, Modified On, Created at, loss, val_loss, accuracy, and Final Test. The first two rows are highlighted with a red box. A red arrow points to the 'Analyze' button in the top right corner of the table.

Name	Run Group	Modified On	Created at	loss	val_loss	accuracy	Final Test
Segmentation-Run-1668f191070...	Default-Run-G...	11/11/22 11:31 AM GMT-8	11/11/22 11:31 AM GMT-8	0.274344032048...	0.044796103990...	0.87707806666...	0.84367377
Segmentation-Run-1668f191070...	Default-Run-G...	11/11/22 10:30 AM GMT-8	11/11/22 10:30 AM GMT-8	0.274344032048...	0.044796103990...	0.87707806666...	0.84367377
Segmentation-Run-1668f191070...	Default-Run-G...	11/11/22 10:30 AM GMT-8	11/11/22 10:30 AM GMT-8	0.274344032048...	0.044796103990...	0.87707806666...	0.84367377
Segmentation-Run-1668f191070...	Default-Run-G...	11/11/22 10:30 AM GMT-8	11/11/22 10:30 AM GMT-8	0.274344032048...	0.044796103990...	0.87707806666...	0.84367377
Segmentation-Run-1668f191070...	Default-Run-G...	11/11/22 10:30 AM GMT-8	11/11/22 10:30 AM GMT-8	0.274344032048...	0.044796103990...	0.87707806666...	0.84367377
Segmentation-Run-1668f191070...	Default-Run-G...	11/11/22 10:30 AM GMT-8	11/11/22 10:30 AM GMT-8	0.274344032048...	0.044796103990...	0.87707806666...	0.84367377
Segmentation-Run-1668f191070...	Default-Run-G...	11/11/22 10:30 AM GMT-8	11/11/22 10:30 AM GMT-8	0.274344032048...	0.044796103990...	0.87707806666...	0.84367377
Segmentation-Run-1668f191070...	Default-Run-G...	11/11/22 10:30 AM GMT-8	11/11/22 10:30 AM GMT-8	0.274344032048...	0.044796103990...	0.87707806666...	0.84367377
Segmentation-Run-1668f191070...	Default-Run-G...	11/11/22 10:30 AM GMT-8	11/11/22 10:30 AM GMT-8	0.274344032048...	0.044796103990...	0.87707806666...	0.84367377
Segmentation-Run-1668f191070...	Default-Run-G...	11/11/22 10:30 AM GMT-8	11/11/22 10:30 AM GMT-8	0.274344032048...	0.044796103990...	0.87707806666...	0.84367377

Figura 2. Ejemplo de listado de *trials* en un experimento. Fuente: AWS, s. f.

4.6. Sesgo y explicabilidad de modelos con SageMaker Clarify

Conceptos generales de sesgo y explicabilidad

Existen dos tipos de sesgos: **de datos**, que se refiere a la infrarrepresentación de determinados grupos o categorías en la muestra; o **de modelos**, en el que un modelo genera una predicción favorable a un determinado grupo o una predicción desfavorable o errónea a otro grupo, sin otra justificación que la pertenencia de las observaciones a estos grupos. Por ejemplo, existe sesgo cuando un modelo deniega la concesión de un crédito a un cliente basándose en su nacionalidad. La existencia de sesgo en unos datos o en un modelo se puede cuantificar a partir de una colección de métricas definidas con este fin.

Por otro lado, la **explicabilidad de un modelo** se refiere a la capacidad de un ser humano de entender cómo ha llegado el modelo a calcular una determinada predicción. Para ello, se utilizan mecanismos de análisis de importancia de características. **SHAP** es uno de estos mecanismos. Cuantifica la influencia de cada característica en una predicción concreta, identificando aquellas que han dirigido más fuertemente la predicción hacia un determinado valor.

Clarify es la herramienta de SageMaker para detectar sesgos y explicar modelos. Puede utilizarse desde la interfaz web a través de Data Wrangler o desde el SDK de SageMaker para Python. También se integra con SageMaker Experiments para registrar la información de sesgo y explicabilidad en los *trials* de un experimento, y con SageMaker Model Monitor para identificar sesgos en datos de producción e incorporar explicabilidad a modelos ya desplegados.

Detección de sesgos en SageMaker

En el siguiente ejemplo se utiliza Clarify desde la SDK de Python para detectar **sesgo pre-training**, es decir, sesgo en los datos.

- ▶ 1. Inicializamos Clarify , especificando la infraestructura de computación que se utilizará para correr el proceso de detección de sesgo:

```
from sagemaker import clarify

procesador_clarify = clarify.SageMakerClarifyProcessor(

    role=role,

    instance_count=1,

    instance_type='ml.m5.xlarge',

    sagemaker_session=sesión)
```

- ▶ 2. A continuación, se especifica la configuración de los datos sobre los que Clarify tratará de identificar sesgo:

```
configuracion_datos = clarify.DataConfig(

    s3_data_input_path='s3://...',

    s3_output_path='s3://...',

    label='característica_objetivo',

    headers=datframe.columns.tolist(),

    dataset_type='text/csv')
```

- ▶ **3.** También es necesario incluir la **configuración del propio proceso de detección de sesgo**, indicando la característica donde creemos que puede haberlo, el valor de esta (grupo) que podría estar perjudicado por el sesgo. Adicionalmente, indicamos el valor de la característica dependiente que corresponde al caso positivo, es decir, aquel respecto del cual puede existir sesgo.

```
configuracion_sesgo = clarify.BiasConfig(  
  
    label_values_or_threshold=['concedido'],  
  
    facet_name=['nacionalidad'],  
  
    facet_values_or_threshold=['española'])
```

- ▶ **4.** El último paso es ejecutar el job de Clarify :

```
clarify_processor.run_pre_training_bias(  
  
    data_config=configuracion_datos,  
  
    data_bias_config=configuracion_sesgo,  
  
    methods='all',  
  
    job_name='job1',  
  
    experiment_config=configuracion_experimento)
```

- 5. Al haber especificado la configuración de un experimento, los resultados de Clarify pueden consultarse en la sección «Experiment and trials». En particular, el informe de sesgo se encuentra en la sección «Bias report» de cada trial.

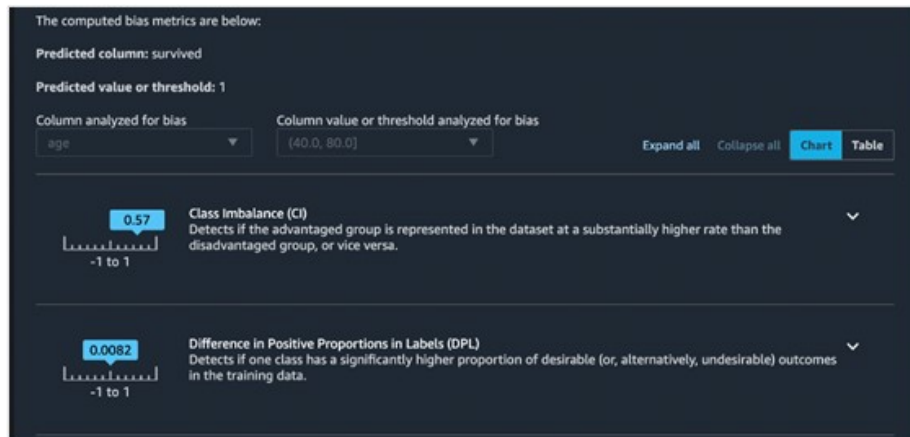


Figura 3. Ejemplo de métricas de sesgo en «Bias Report» de Sage Maker. Fuente: AWS, s. f.

La detección de sesgo en un modelo sigue un procedimiento similar. La principal diferencia es que tenemos que especificar, adicionalmente, una configuración relacionada con el modelo. En esta configuración se incluye el **nombre del modelo** sobre el que se pretende identificar sesgo. También es necesario especificar el **valor de probabilidad** a partir del cual se asigna la clase positiva de la predicción.

```
model_config = clarify.ModelConfig(

    model_name='modelo1',

    instance_type='ml.m5.xlarge',

    instance_count=1,

    accept_type='text/csv',

    content_type='text/csv')

configuracion_objetivo=clarify.ModelPredictedLabelConfig(probability_threshold=0.5)
```

Como en el caso anterior, la información de los resultados se almacena en la sección «Bias report» del *trial* correspondiente.

Explicabilidad de modelos en SageMaker con SHAP

Para el cálculo de la importancia de características, SageMaker utiliza **SHAP**, un algoritmo que calcula el **valor Shapley** de cada característica. Este valor representa el grado de influencia de la característica en una predicción determinada.

En el siguiente ejemplo se muestra cómo calcular la importancia de características con Clarify a través del SDK de Python:

- ▶ **1.** En primer lugar, se define la ubicación de los datos y su formato, el nombre de las características del dataset y cuál de ellas es la característica objetivo:

```
configuracion_explicabilidad = clarify.DataConfig(

    s3_data_input_path='s3://',

    s3_output_path='s3://...',

    label='característica_objetivo',

    headers=df_sampled.columns.tolist(),

    dataset_type='text/csv')
```

- ▶ **2.** A continuación, se define un *baseline*. Un *baseline* en SHAP es una instancia de datos sintética que representa tanto como sea posible al conjunto de observaciones reales del dataset. En este caso, elegimos el valor de moda (valor más frecuente) de cada característica, que entendemos que es la mejor representación del conjunto de datos:

```
baseline = df.mode().iloc[0, 1:].astype(int).tolist()
```

- ▶ **3.** El siguiente paso es especificar la configuración de SHAP . Indicamos el baseline construido en el paso anterior y el método que se utilizará para calcular la influencia global de cada característica a partir de la influencia en cada predicción individual. En este caso, elegimos el promedio del valor absoluto de cada valor Shapley calculado:

```
configuracion_shap = clarify.SHAPConfig(  
  
baseline=[baseline],  
  
agg_method='mean_abs')
```

- ▶ **4.** Por último, ejecutamos el job de Clarify para generar los resultados:

```
clarify_processor.run_explainability(  
  
    data_config= configuracion_explicabilidad,  
  
    model_config=configuracion_modelo,  
  
    explainability_config= configuracion_shap,  
  
    job_name='job1',  
  
    experiment_config=configuracion_experimento,  
  
    wait=False,  
  
    logs=False)
```


- 5. Los resultados obtenidos pueden consultarse en la sección «Explainability» del *trial* correspondiente.

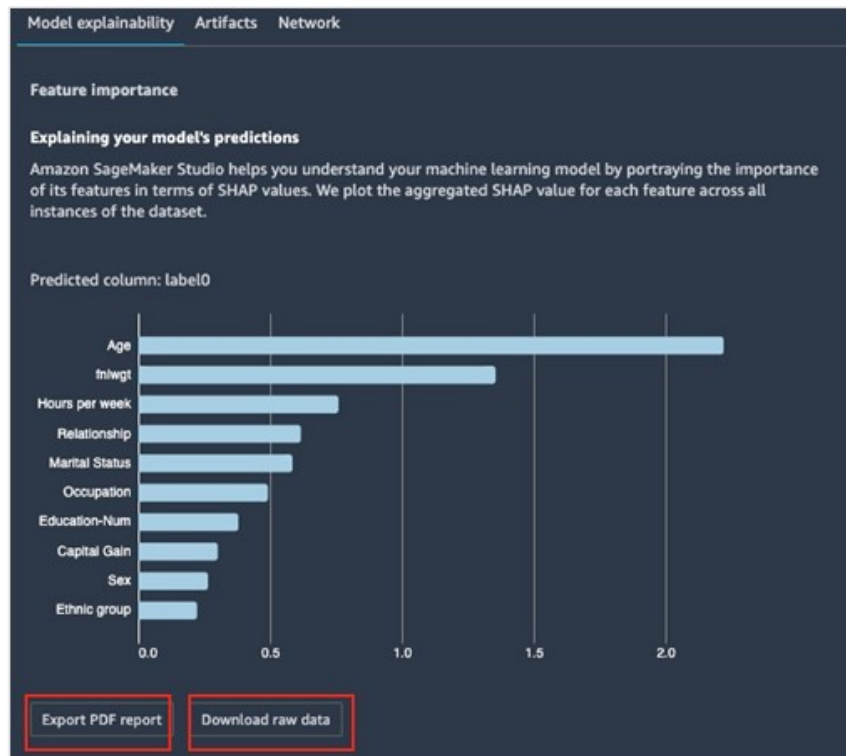
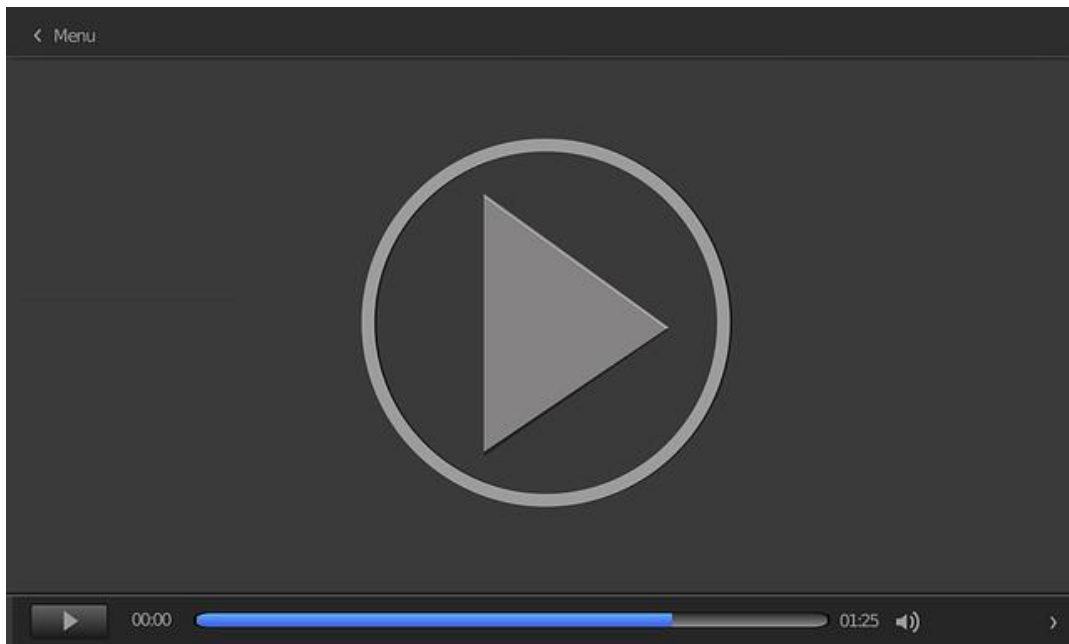


Figura 4. Resultados del proceso de detección de importancia de características en SageMaker. Fuente: Teh, 2021.

En el siguiente vídeo, *Uso de SageMaker Clarify para reducción del sesgo e interpretabilidad de predicciones*, se muestra un ejemplo real de uso de la herramienta Clarify en SageMaker.



Accede al vídeo:

<https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=368627ab-1540-4c84-9123-b149011dc0ff>

4.7. Monitorización y mantenimiento de modelos con herramientas de SageMaker

Consideraciones generales

La monitorización de los modelos es una actividad clave para detectar **cambios en las características de los datos** (*concept drift*) que se procesan en el entorno de producción. Estos cambios suelen provocar que el rendimiento de los modelos se degrade considerablemente.

Para identificar la degradación del rendimiento, pueden utilizarse dos enfoques:

- ▶ Comparación de las predicciones en producción con los **valores reales** o *ground truth*.
- ▶ En caso de que no estén disponibles los valores reales, se puede comparar la **distribución estadística de los datos**, el momento actual respecto a la obtenida en el proceso de entrenamiento. Si existen discrepancias, es esperable que el modelo no retorne predicciones precisas.

SageMaker Model Monitor proporciona funcionalidades para medir la calidad de las predicciones con los dos enfoques anteriores. Además, utiliza Clarify para detectar sesgos en los datos/modelos desplegados en producción y explicar las predicciones generadas por estos modelos.

Por defecto, el *endpoint* en el que se despliega un modelo no persiste los resultados de la inferencia. Puede habilitarse esta opción de la siguiente forma:

```
configuracion_captura_datos = DataCaptureConfig(  
  
    enable_capture=True,  
  
    sampling_percentage=100,
```

```
destination_s3_uri='s3://...')
```

Creación de un monitor de calidad de datos

Para crear un monitor de calidad de datos basado en *baseline* (es decir, en las características de los datos observadas en el pasado), deben ejecutarse los pasos que se indican a continuación en SageMaker:

- ▶ Seleccionar «Endpoints» en la sección «SageMaker Resources».
- ▶ Seleccionar el *endpoint* en el que está desplegado el modelo cuyos datos queremos monitorizar.
- ▶ En la sección «Data quality», hacer clic en «Create monitoring schedule».
- ▶ En el cuadro de diálogo que se muestra a continuación, especificar el tiempo máximo de ejecución del *job* de monitorización (*stopping condition*) y su frecuencia (*schedule expression*), así como la infraestructura de computación que utilizará el *job* y la ruta S3 en la que se almacenarán los resultados del proceso de monitorización.
- ▶ Finalmente, especificar la ruta S3 a los datos de entrenamiento, y otra ruta S3 para almacenar la información del *baseline* construido a partir de ellos.
- ▶ Una vez finalizada la configuración, el *job* comienza a ejecutarse con la frecuencia especificada.
- ▶ Si se detecta un cambio significativo en la distribución estadística de los datos, se mostrará el mensaje «Issue found» en la columna Monitoring status de la sección Monitoring job history.

MODEL MONITORING

Endpoint: DEMO-xgb-churn-model-quality-monitor-2021-10-14-1616

Test inference

Data quality

Model quality

Model explainability

Model bias

Monitoring job history

AWS settings

Monitoring status	Monitoring job name	Monitoring schedule name	Monitoring job type	Created
No issues	model-quality-monitoring-2021...	DEMO-xgb-churn-monitoring-sc...	ModelQuality	55 minutes ago
Issue found	model-quality-monitoring-2021...	DEMO-xgb-churn-monitoring-sc...	ModelQuality	2 hours ago
Issue found	model-quality-monitoring-2021...	DEMO-xgb-churn-monitoring-sc...	ModelQuality	3 hours ago
No issues	model-quality-monitoring-2021...	DEMO-xgb-churn-monitoring-sc...	ModelQuality	4 hours ago

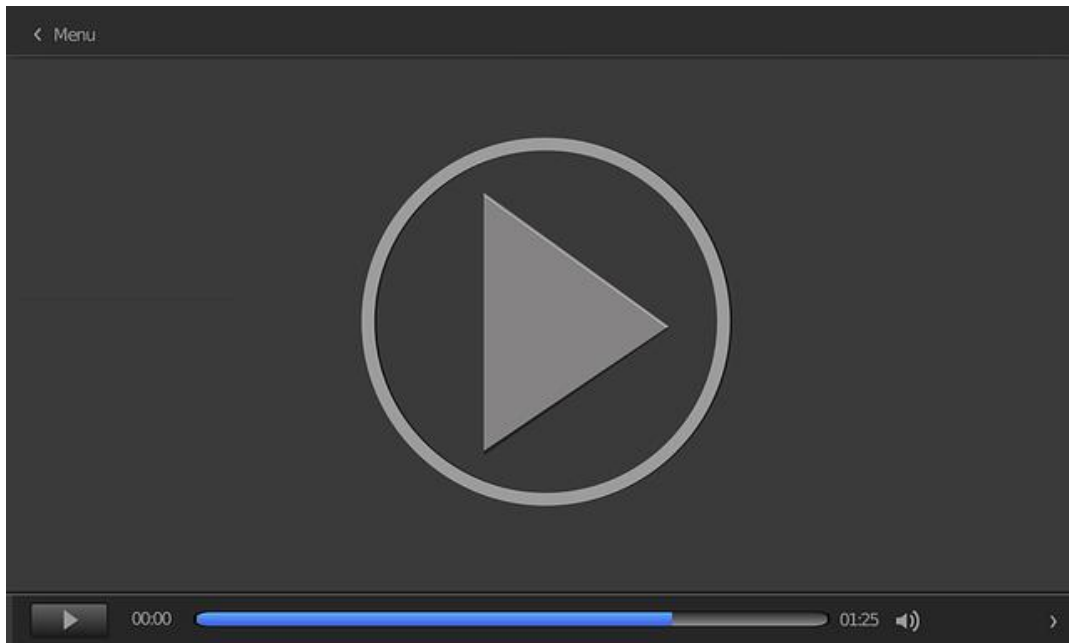
Figura 5. Ejemplo de histórico de *jobs* de monitorización de datos y modelos. Fuente: Nigenda, Karnin, Zafar, Ramesha, Tan, Donini y Kenthapadi, 2021.

Creación de un monitor de calidad de modelos

Para crear un **monitor de calidad de modelos** basado en datos de *ground truth* en SageMaker, deben ejecutarse los pasos que se indican a continuación:

- ▶ Ejecutar los pasos 1-3 de la sección anterior.
- ▶ Especificar la infraestructura en la que se ejecutará el *job* de monitorización, la ruta S3 para almacenar los resultados y la ubicación en S3 de la información de *ground truth*. Es posible indicar también el *delay* temporal que existe entre las predicciones y la disponibilidad de los datos de *ground truth* correspondientes.
- ▶ En el último paso se debe especificar el atributo de los datos de *ground truth* que contiene el valor real y el atributo de la inferencia que contiene la predicción para poder compararlos.
- ▶ Como en el caso anterior, el *job* comienza a ejecutarse con la periodicidad especificada.
- ▶ Los resultados pueden consultarse en la sección «Monitoring job history», como en el caso anterior.

Adicionalmente, en el siguiente vídeo, *Implementación de A/B testing en endpoints*, se muestra un ejemplo real de comparación del rendimiento de dos modelos en producción a través de *A/B testing*.



Accede al vídeo:

<https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=42e77e93-6c62-4b0e-affc-b14c01083049>

4.8. Cuaderno de ejercicios

1. Escribe el código fuente en Python necesario para entrenar en SageMaker un modelo con MXNet 1.2.1 en una única instancia de tipo `ml.p2.xlarge`, a partir del contenido del *script* `train.py` y datos de entrenamiento disponibles en un bucket S3. A continuación, despliega el modelo en un endpoint en una `ml.p2.xlarge` y utilízalo para obtener predicciones sobre datos de prueba. Por último, elimina el endpoint y el modelo asociado.

```
from sagemaker.mxnet import MXNet

mxnet_estimator = MXNet('train.py',

                        role='SageMakerRole',

                        instance_type='ml.p2.xlarge',

                        instance_count=1,

                        framework_version='1.2.1')

mxnet_estimator.fit('s3://my_bucket/my_training_data/')

mxnet_predictor = mxnet_estimator.deploy(initial_instance_count=1,
instance_type='ml.p2.xlarge')

response = mxnet_predictor.predict(test_data)

mxnet_predictor.delete_endpoint()

mxnet_predictor.delete_model()
```

2. Utilizando la clase `HyperparameterTuner` de SageMaker, escribe el código fuente en Python para calcular el valor óptimo de `learning rate` en términos de `validation accuracy`, a partir de un estimador existente (`my_estimator`). El número máximo de `jobs` totales y concurrentes debe ser 100. A continuación, despliega el modelo óptimo y utilízalo para obtener predicciones de unos datos de prueba (`my_prediction_data`).

```
from sagemaker.tuner import HyperparameterTuner, ContinuousParameter

my_tuner = HyperparameterTuner(

    estimator=my_estimator,

    objective_metric_name='validation-accuracy',

    hyperparameter_ranges={'learning-rate':
ContinuousParameter(0.05, 0.06)},

    metric_definitions=[{'Name': 'validation-accuracy', 'Regex':
'validation-accuracy=(\d\.\d+)'}],

    max_jobs=100,

    max_parallel_jobs=10)

my_tuner.fit({'train': 's3://my_bucket/my_training_data', 'test':
's3://my_bucket_my_testing_data'})

my_predictor = my_tuner.deploy(initial_instance_count=1,
instance_type='ml.m4.xlarge')

response = my_predictor.predict(my_prediction_data)
```


3. Utilizando `JumpStart` , escribe el código fuente en Python para desplegar en un endpoint el modelo de Hugging Face `huggingface-text2text-flan-t5-xl` , indicando que las peticiones de predicciones deben tener formato JSON.

```
from sagemaker.jumpstart.model import JumpStartModel

model_id = "huggingface-text2text-flan-t5-xl"

my_model = JumpStartModel(model_id=model_id)

base_model_predictor = my_model.deploy()

base_model_predictor.accept = "application/json;verbose"
```

4. Utilizando las clases `MXNetModel` y `ServerlessInferenceConfig` de SageMaker, escribe el código fuente en Python para desplegar un modelo MXNet (1.6.0) preentrenado almacenado en bucket S3 , cuyo script de inferencia es `inference.py` . El despliegue debe ser de tipo `serverless` , con una memoria de 4096 MB y concurrencia máxima de 10 invocaciones.

```
from sagemaker.mxnet import MXNetModel

from sagemaker.serverless import ServerlessInferenceConfig

import sagemaker

role = sagemaker.get_execution_role()

model = MXNetModel(

    model_data="s3://my_bucket/pretrained_model/model.tar.gz",

    role=role,

    entry_point="inference.py",

    py_version="py3",
```

```

        framework_version="1.6.0",
    )

    from sagemaker.serverless import ServerlessInferenceConfig

    serverless_config = ServerlessInferenceConfig(

        memory_size_in_mb=4096,

        max_concurrency=10,
    )

    serverless_predictor =
    model.deploy(serverless_inference_config=serverless_config)

```

5. Utilizando el SDK de SageMaker, escribe el código fuente en Python para crear un nuevo experimento con el nombre AutoML, y un nuevo *trial* dentro del experimento, con el nombre `random-forest`. A continuación, imprime en pantalla la lista de experimentos y la lista de *trials* en el experimento que acabas de crear. Finalmente, elimina el experimento.

```

from smexperiments import experiment

my_experiment = experiment.Experiment.create(experiment_name='AutoML')

my_trial = my_experiment.create_trial(trial_name='random-forest')


for exp in experiment.Experiment.list():

    print(exp)

for trial in my_experiment.list_trials():

    print(trial)

```

```
my_experiment.delete_all(action="--force")
```

4.9. Referencias bibliográficas

AWS (s. f.). *Create, store, and share features with Feature Store*.
<https://docs.aws.amazon.com/sagemaker/latest/dg/feature-store.html>

AWS (s. f.). *Generate Reports for Bias in Pre-training Data in SageMaker Studio*.
<https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-data-bias-reports-ui.html>

AWS (s. f.). *What is Amazon SageMaker?*
<https://docs.aws.amazon.com/sagemaker/latest/dg/experiments-view-compare.html>

Ma, C., Govindaram, A. y Nguyen, H. (2023, noviembre 29). Accelerate data preparation for ML in Amazon SageMaker Canvas. AWS.
<https://aws.amazon.com/es/blogs/machine-learning/accelerate-data-preparation-for-ml-with-comprehensive-data-preparation-capabilities-and-a-natural-language-interface-in-amazon-sagemaker-canvas/>

Nigenda, D., Karnin, Z., Zafar, M. B., Ramesha, R., Tan, A., Donini, M. y Kenthapadi, K. (2021). Amazon sagemaker model monitor: A system for real-time insights into deployed machine learning models. En *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (pp. 3671-3681).

Teh, W. (2021, noviembre 10). Use integrated explainability tools and improve model quality using Amazon SageMaker Autopilot. AWS.
<https://aws.amazon.com/es/blogs/machine-learning/use-integrated-explainability-tools-and-improve-model-quality-using-amazon-sagemaker-autopilot/>

Amazon SageMaker examples

AWS (s. f.). *Amazon-sagemaker-examples* [Conjunto de datos]. GitHub. <https://github.com/aws/amazon-sagemaker-examples>

El repositorio de Github oficial de SageMaker proporciona una colección de ejemplos que profundizan en el uso de las herramientas vistas en este tema. Se incluyen tutoriales y *notebooks* Jupyter, en los que se detalla el uso del SDK de Python y la interfaz web (SageMaker Studio). Estos recursos permiten aprender a explotar las principales funcionalidades disponibles para resolver los casos de uso más habituales en diferentes sectores de actividad.

Amazon SageMaker documentation

AWS (s. f.). *Amazon sagemaker documentation*.
<https://docs.aws.amazon.com/sagemaker/>

La documentación oficial de AWS SageMaker constituye un recurso muy valioso para ampliar el conocimiento de cada una de las herramientas vistas en este tema, así como para profundizar en los casos prácticos que permiten adquirir la experiencia necesaria para trabajar con SageMaker de forma autónoma.

Por otro lado, la documentación oficial es la que refleja con mayor rapidez los cambios que se producen en cada una de las herramientas, por lo que permiten mantenerse actualizado respecto a nuevas funcionalidades que se incorporan.

1. ¿Para qué sirve la herramienta JumpStart de SageMaker?
 - A. JumpStart es la infraestructura de computación gestionada por Amazon, que podemos utilizar para ejecutar procesos de entrenamiento.
 - B. JumpStart es un catálogo de soluciones y modelos que resuelven los casos de uso más habituales de cada sector.
 - C. JumpStart es una colección de *datasets* proporcionados por Amazon para realizar pruebas de algoritmos y establecer *benchmarks*.
 - D. JumpStart es la solución de Amazon que permite detectar sesgos e importancia de características en una predicción.

2. ¿Qué datos se pueden almacenar en cada *trial* de un experimento?
 - A. Solo la información relativa al experimento, es decir, su nombre y su descripción.
 - B. Datos empleados, transformaciones aplicadas, valores de hiperparámetros, estimador elegido, métricas de rendimiento y gráficas generadas.
 - C. Datos empleados, transformaciones aplicadas y valores de hiperparámetros.
 - D. Datos empleados, transformaciones aplicadas, valores de hiperparámetros, estimador elegido y métricas de rendimiento.

3. ¿Qué efecto tiene la llamada `sagemaker.Session()` ?
 - A. Devuelve información acerca de la sesión SageMaker en curso.
 - B. Crea una sesión de SageMaker, a través de la cual se utilizan todas las funcionalidades que ofrece el SDK.
 - C. No tiene efecto, puesto que el *notebook* ya dispone de una sesión que puede utilizar para realizar llamadas al SDK.
 - D. Crea una sesión de SageMaker, que posteriormente tenemos que activar con la llamada `enable()` para acceder a las funcionalidades del SDK.

4. ¿Para qué puede utilizarse en Data Wrangler un *scatter plot* que enfrenta los valores de dos características?
- A. No es posible representar los valores de dos características en un *scatter plot*.
 - B. Para detectar correlación entre las características.
 - C. Para determinar la frecuencia de cada valor o rango de valores de cada una de las características.
 - D. Data Wrangler no ofrece la posibilidad de visualizar un *scatter plot*.
5. ¿Qué es el modelado rápido de Data Wrangler?
- A. Un ejercicio de construcción de un modelo con el perfil de *hardware* más potente del que dispone Amazon.
 - B. Un ejercicio de construcción de un modelo a partir de un fragmento de los datos para anticipar la viabilidad de un modelo completo.
 - C. Un ejercicio de construcción de un modelo con GPUs, que son el *hardware* óptimo para acelerar el proceso de entrenamiento.
 - D. Un ejercicio de construcción de un modelo con un algoritmo trivial, lo que permite obtener resultados aproximados rápidamente.
6. ¿Cuáles son los tres componentes de un Feature Store?
- A. Experimentos, definiciones de características y datos ingestados.
 - B. Colección de fuentes de datos de las que extrae características y datos, definiciones de características y datos ingestados.
 - C. Colección de fuentes de datos de las que extrae características y datos, definiciones de características e hiperparámetros.
 - D. Colección de fuentes de datos de las que extrae características y datos, modelos e hiperparámetros.

7. ¿Qué implicaciones positivas tiene la centralización de características en un único repositorio?
- A. Permite que se utilicen las mismas características en el entrenamiento y en la inferencia.
 - B. Evita que varios científicos de datos construyan las mismas características por separado, y permite que se utilicen las mismas características en el entrenamiento y en la inferencia.
 - C. Evita que varios científicos de datos construyan las mismas características por separado.
 - D. Evita que cada científico de datos construya su propio *feature store*.
8. ¿Para qué sirve la llamada `sagemaker.image_uris.retrieve()` ?
- A. Retorna un puntero a un contenedor que incluye los paquetes de *software* correspondientes a un *framework*, como TensorFlow o PyTorch.
 - B. Retorna un puntero a una imagen que contiene los paquetes de *software* correspondientes a un *framework*, como TensorFlow o PyTorch.
 - C. Retorna un puntero a una imagen que contiene un sistema operativo optimizado para la creación de modelos.
 - D. Retorna el listado de imágenes disponibles para encapsular procesos de preparación de datos o entrenamiento.

9. ¿Qué ventaja tiene incorporar los *notebooks* a un repositorio Git?
- A. Que se dispone de un *backup* en caso de que se borren accidentalmente las copias originales.
 - B. La posibilidad de identificar fácilmente los cambios que se han realizado de una versión a otra.
 - C. No presenta ninguna ventaja, simplemente es una práctica habitual en el ámbito DevOps.
 - D. La posibilidad de identificar fácilmente las diferencias entre dos *notebooks* similares.
10. ¿En qué consiste el sesgo de modelos?
- A. En que un modelo genera predicciones erróneas, porque no disponía de suficientes datos durante el proceso de entrenamiento.
 - B. En que un modelo genera una predicción favorable a un grupo o desfavorable o errónea a otro grupo, por la simple pertenencia a estos grupos.
 - C. En que un modelo genera una predicción favorable a un grupo o desfavorable o errónea a otro grupo, basándose en múltiples criterios objetivos.
 - D. El sesgo solo existe en los datos, no en los modelos.