MLflow

NetApp Solutions

NetApp
December 19, 2024

This PDF was generated from https://docs.netapp.com/es-es/netapp-solutions/ai/aicp_mlflow_deployment.html on December 19, 2024. Always check docs.netapp.com for the latest.

Tabla de contenidos

MLflow

Despliegue de MLflow

Esta sección describe las tareas que debe completar para implementar MLflow en su clúster de Kubernetes.

Es posible poner en marcha MLflow en plataformas que no sean Kubernetes. La implementación de MLflow en plataformas distintas de Kubernetes no está incluida en el alcance de esta solución.

Requisitos previos

Antes de realizar el ejercicio de implementación descrito en esta sección, asumimos que ya ha realizado las siguientes tareas:

1. Ya tiene un clúster de Kubernetes en funcionamiento.
2. Ya ha instalado y configurado NetApp Trident en su clúster de Kubernetes. Para obtener más información sobre Trident, consulte la "Documentación de Trident".

Instale el Helm

MLflow se implementa usando Helm, un popular administrador de paquetes para Kubernetes. Antes de implementar MLflow, debes instalar Helm en tu nodo de control de Kubernetes. Para instalar Helm, siga el "instrucciones de instalación" en la documentación oficial de Helm.

Establezca el tipo de almacenamiento de Kubernetes predeterminado

Antes de implementar MLflow, debe designar un StorageClass predeterminado dentro de su clúster de Kubernetes. Para designar una clase de almacenamiento predeterminada en su clúster, siga las instrucciones descritas en la "Despliegue de Kubeflow" sección. Si ya ha designado un tipo de almacenamiento predeterminado en el clúster, puede omitir este paso.

Desplegar MLflow

Una vez que se hayan cumplido los requisitos previos, puede comenzar con el despliegue de MLflow utilizando el diagrama de timón.

Configurar el despliegue de gráficos de Helm de MLflow.

Antes de implementar MLflow usando el diagrama Helm, podemos configurar la implementación para usar la clase de almacenamiento de NetApp Trident y cambiar otros parámetros para satisfacer nuestras necesidades utilizando un archivo **config.yaml**. Un ejemplo de archivo **config.yaml** se puede encontrar en: https://github.com/bitnami/charts/blob/main/bitnami/mlflow/values.yaml

Puede establecer la clase de almacenamiento de Trident con el parámetro **global.defaultStorageClass** en el archivo config.yaml (por ejemplo, storageClass: «ontap-flexvol»).

Instalación de la tabla Helm Chart

El diagrama Helm se puede instalar con el archivo personalizado **config.yaml** para MLflow usando el siguiente comando:

```
helm install oci://registry-1.docker.io/bitnamicharts/mlflow -f
config.yaml --generate-name --namespace jupyterhub
```

El comando implementa MLflow en el clúster de Kubernetes en la configuración personalizada a través del archivo **config.yaml** proporcionado. MLflow se implementa en el espacio de nombres dado y se da un nombre de versión aleatorio a través de kubernetes para el lanzamiento.

Comprobar despliegue

Una vez que el diagrama de Helm haya terminado de desplegarse, puede comprobar si el servicio es accesible mediante:

```
kubectl get service -n jupyterhub
```

Reemplace **jupyterhub** con el espacio de nombres que utilizó durante la implementación.

Deberías ver los siguientes servicios:

```
NAME                                TYPE        CLUSTER-IP      EXTERNAL-IP
PORT(S)             AGE
mlflow-1719843029-minio             ClusterIP   10.233.22.4     <none>
80/TCP,9001/TCP     25d
mlflow-1719843029-postgresql        ClusterIP   10.233.5.141    <none>
5432/TCP            25d
mlflow-1719843029-postgresql-hl     ClusterIP   None            <none>
5432/TCP            25d
mlflow-1719843029-tracking          NodePort    10.233.2.158    <none>
30002:30002/TCP     25d
```

Editamos el archivo config.yaml para usar el servicio NodePort para acceder a MLflow en el puerto 30002.

Acceder a MLflow

Una vez que todos los servicios relacionados con MLflow estén activos y en ejecución, puede acceder a ellos utilizando la dirección IP de NodePort o LoadBalancer (por ejemplo `http://10.61.181.109:30002`)

Trazabilidad de conjunto de datos a modelo con NetApp y MLflow

Se "Kit de herramientas Data OPS de NetApp para Kubernetes" puede utilizar junto con las capacidades de seguimiento de experimentos de MLflow para implementar la trazabilidad de código a conjunto de datos, conjunto de datos a modelo o espacio de trabajo a modelo.

En el cuaderno de ejemplo se utilizaron las siguientes bibliotecas:

Requisitos previos

1. "rayos de la antorcha"
2. "cuda_runtime"
3. "cudnn"
4. "tritón"
5. "Kit de herramientas de DataOps de NetApp para Kubernetes"

Para implementar la trazabilidad de modelo de conjunto de datos de código o de espacio de trabajo a modelo, simplemente cree una instantánea de su conjunto de datos o volumen de espacio de trabajo utilizando el kit de herramientas de DataOps como parte de su ciclo de entrenamiento, como se muestra en el siguiente fragmento de código de ejemplo. Este código guardará el nombre del volumen de datos y el nombre de la instantánea como etiquetas asociadas a la ejecución de entrenamiento específica que está registrando en el servidor de seguimiento de experimentos MLflow.

```
...
from netapp_dataops.k8s import cloneJupyterLab, createJupyterLab,
deleteJupyterLab, \
listJupyterLabs, createJupyterLabSnapshot, listJupyterLabSnapshots,
restoreJupyterLabSnapshot, \
cloneVolume, createVolume, deleteVolume, listVolumes,
createVolumeSnapshot, \
deleteVolumeSnapshot, listVolumeSnapshots, restoreVolumeSnapshot


mlflow.set_tracking_uri("<your_tracking_server_uri>>:<port>>")
    os.environ['MLFLOW_HTTP_REQUEST_TIMEOUT'] = '500'  # Increase to 500
seconds
    mlflow.set_experiment(experiment_id)
    with mlflow.start_run() as run:
        latest_run_id = run.info.run_id
        start_time = datetime.now()

        # Preprocess the data
        preprocess(input_pdf_file_path, to_be_cleaned_input_file_path)
```

```
        # Print out sensitive data (passwords, SECRET_TOKEN, API_KEY
found)
        check_pretrain(to_be_cleaned_input_file_path)

        # Tokenize the input file
        pretrain_tokenization(to_be_cleaned_input_file_path, model_name,
tokenized_output_file_path)

        # Load the tokenizer and model
        tokenizer = GPT2Tokenizer.from_pretrained(model_name)
        model = GPT2LMHeadModel.from_pretrained(model_name)

        # Set the pad token
        tokenizer.pad_token = tokenizer.eos_token
        tokenizer.add_special_tokens({'pad_token': '[PAD]'})

        # Encode, generate, and decode the text
        with open(tokenized_output_file_path, 'r', encoding='utf-8') as
file:
            content = file.read()
        encode_generate_decode(content, decoded_output_file_path,
tokenizer=tokenizer, model=model)

        # Save the model
        model.save_pretrained(model_save_path)
        tokenizer.save_pretrained(model_save_path)

        # Finetuning here
        with open(decoded_output_file_path, 'r', encoding='utf-8') as
file:
            content = file.read()
        model.finetune(content, tokenizer=tokenizer, model=model)

        # Evaluate the model using NLTK
        output_set = Dataset.from_dict({"text": [content]})
        test_set = Dataset.from_dict({"text": [content]})
        scores = nltk_evaluation_gpt(output_set, test_set, model=model,
tokenizer=tokenizer)
        print(f"Scores: {scores}")

        # End time and elapsed time
        end_time = datetime.now()
        elapsed_time = end_time - start_time
        elapsed_minutes = elapsed_time.total_seconds() // 60
        elapsed_seconds = elapsed_time.total_seconds() % 60

        # Create DOTK snapshots for code, dataset, and model
```

```
        snapshot = createVolumeSnapshot(pvcName="model-pvc",
namespace="default", printOutput=True)

        #Log snapshot IDs to MLflow
        mlflow.log_param("code_snapshot_id", snapshot)
        mlflow.log_param("dataset_snapshot_id", snapshot)
        mlflow.log_param("model_snapshot_id", snapshot)

        # Log parameters and metrics to MLflow
        mlflow.log_param("wf_start_time", start_time)
        mlflow.log_param("wf_end_time", end_time)
        mlflow.log_param("wf_elapsed_time_minutes", elapsed_minutes)
        mlflow.log_param("wf_elapsed_time_seconds", elapsed_seconds)

        mlflow.log_artifact(decoded_output_file_path.rsplit('/', 1)[0])  #
Remove the filename to log the directory
        mlflow.log_artifact(model_save_path) # log the model save path

        print(f"Experiment ID: {experiment_id}")
        print(f"Run ID: {latest_run_id}")
        print(f"Elapsed time: {elapsed_minutes} minutes and
{elapsed_seconds} seconds")
```

El fragmento de código anterior registra los ID de instantánea en el servidor de seguimiento de experimentos MLflow, que se puede utilizar para rastrear el conjunto de datos y el modelo específicos que se utilizaron para entrenar el modelo. Esto le permitirá rastrear el conjunto de datos y el modelo específicos que se utilizaron para entrenar el modelo, así como el código específico que se utilizó para preprocesar los datos, tokenizar el archivo de entrada, cargar el tokenizador y el modelo, codificar, generar y descodificar el texto, guardar el modelo, afinar el modelo, evaluar el modelo utilizando "NLTK"puntuaciones de perplejidad y registrar los hiperparámetros y métricas para fluir. Por ejemplo, la siguiente figura muestra el error medio cuadrado (MSE) de un modelo scikit-learn para diferentes ejecuciones de experimentos:

[MSEs. Mlrun mlflow sklean MLmodels aicp] | *aicp_mlrun-mlflow_sklean-MLmodels_MSEs.png*

Es sencillo para el análisis de datos, los propietarios de líneas de negocio y los ejecutivos comprender e inferir qué modelo funciona mejor bajo sus restricciones particulares, configuraciones, período de tiempo y otras circunstancias. Para obtener más detalles sobre cómo preprocesar, tokenizar, cargar, codificar, generar, decodificar, guardar, ajustar y evaluar el modelo, consulte el `dotk-mlflow` ejemplo de Python empaquetado en el `netapp_dataops.k8s` repositorio.

Para obtener más información sobre cómo crear instantáneas de su conjunto de datos o espacio de trabajo JupyterLab, consulte la "Kit de herramientas de DataOps de NetApp".

En cuanto a los modelos que fueron entrenados, se utilizaron los siguientes modelos en el cuaderno dotk-mlflow:

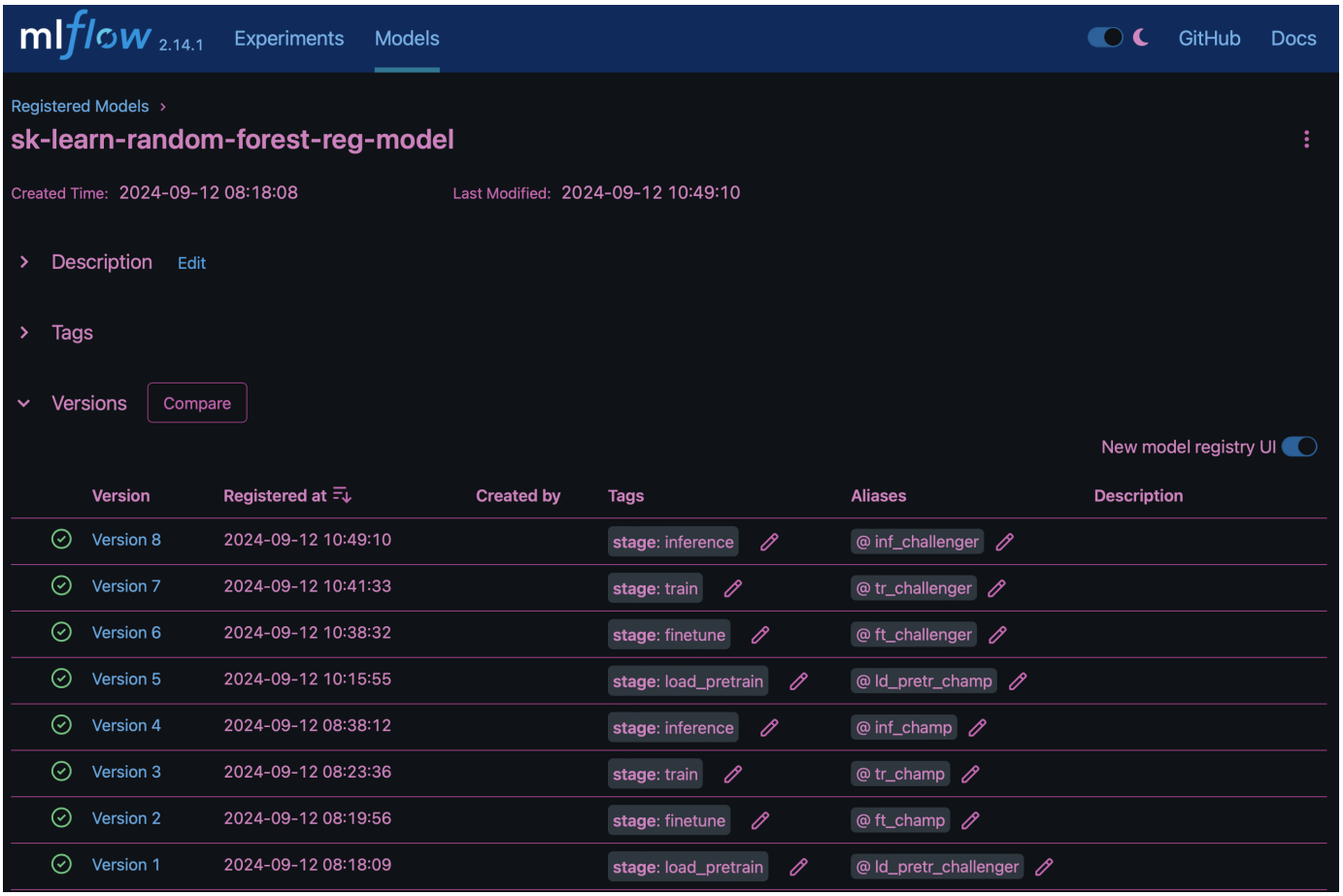
Modelos

1. "GPT2LMHeadModel": El transformador modelo GPT2 con una cabeza de modelado de lenguaje en la parte superior (capa lineal con pesos atados a las incrustaciones de entrada). Es un modelo de

transformador que ha sido pre-entrenado en un gran corpus de datos de texto y finetuned en un conjunto de datos específico. Utilizamos el modelo GPT2 predeterminado "máscara de atención"para procesar por lotes secuencias de entrada con el tokenizador correspondiente para su modelo de elección.

2. "PHI-2"Phi-2 es un transformador con 2,7 mil millones de parámetros. Se entrenó utilizando las mismas fuentes de datos que Phi-1,5, aumentada con una nueva fuente de datos que consta de varios textos sintéticos de NLP y sitios web filtrados (por seguridad y valor educativo).
3. "XLNet (modelo de tamaño basado)": Modelo XLNet pre-entrenado en idioma inglés. Fue introducido en el artículo "XLNet: Preentrenamiento autorregresivo generalizado para la comprensión del lenguaje" por Yang et al. Y publicado por primera vez en este "repositorio".

El resultado "Registro de modelos en MLflow"contendrá los siguientes modelos, versiones y etiquetas de bosque aleatorio:



Para implementar el modelo en un servidor de inferencia a través de Kubernetes, simplemente ejecute el siguiente Jupyter Notebook. Tenga en cuenta que en este ejemplo `dotk-mlflow`, en lugar de usar el paquete, estamos modificando la arquitectura del modelo de regresión forestal aleatoria para minimizar el error medio-cuadrado (MSE) en el modelo inicial, y por lo tanto creamos múltiples versiones de dicho modelo en nuestro Registro de Modelos.

```
from mlflow.models import Model
mlflow.set_tracking_uri("http://<tracking_server_URI_with_port>")
experiment_id='<your_specified_exp_id>'

# Alternatively, you can load the Model object from a local MLmodel file
```

```
# model1 = Model.load("~/path/to/my/MLmodel")

from sklearn.datasets import make_regression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

import mlflow
import mlflow.sklearn
from mlflow.models import infer_signature

# Create a new experiment and get its ID
experiment_id = mlflow.create_experiment(experiment_id)

# Or fetch the ID of the existing experiment
# experiment_id =
mlflow.get_experiment_by_name("<your_specified_exp_id>").experiment_id

with mlflow.start_run(experiment_id=experiment_id) as run:
    X, y = make_regression(n_features=4, n_informative=2, random_state=0,
shuffle=False)
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42
    )
    params = {"max_depth": 2, "random_state": 42}
    model = RandomForestRegressor(**params)
    model.fit(X_train, y_train)

    # Infer the model signature
    y_pred = model.predict(X_test)
    signature = infer_signature(X_test, y_pred)

    # Log parameters and metrics using the MLflow APIs
    mlflow.log_params(params)
    mlflow.log_metrics({"mse": mean_squared_error(y_test, y_pred)})

    # Log the sklearn model and register as version 1
    mlflow.sklearn.log_model(
        sk_model=model,
        artifact_path="sklearn-model",
        signature=signature,
        registered_model_name="sk-learn-random-forest-reg-model",
    )
```

El resultado de la ejecución de su celda Jupyter Notebook debe ser similar al siguiente, con el modelo registrado como versión `3` en el Registro de modelos:

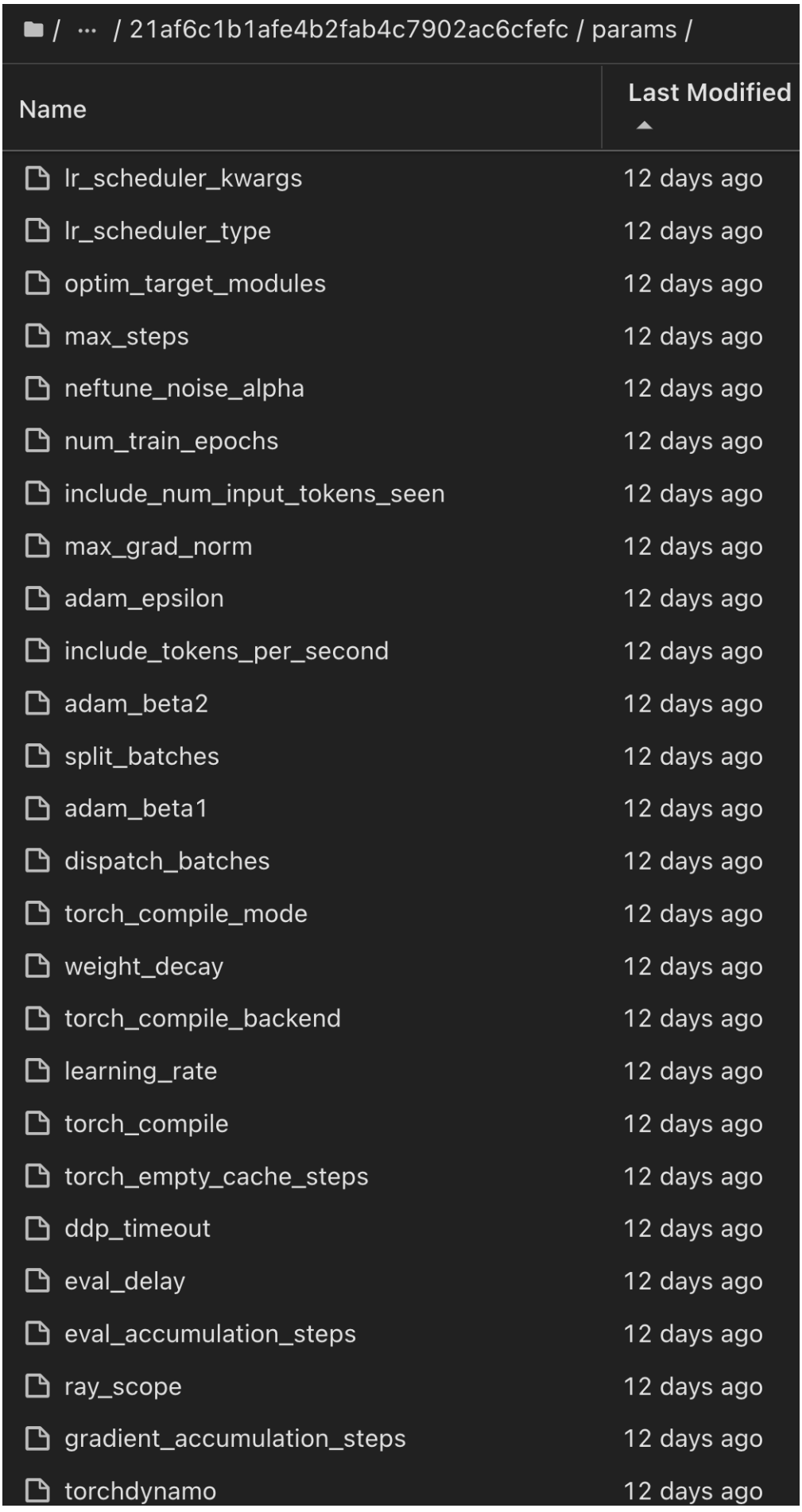
```
Registered model 'sk-learn-random-forest-reg-model' already exists.
Creating a new version of this model...
2024/09/12 15:23:36 INFO mlflow.store.model_registry.abstract_store:
Waiting up to 300 seconds for model version to finish creation. Model
name: sk-learn-random-forest-reg-model, version 3
Created version '3' of model 'sk-learn-random-forest-reg-model'.
```

En el registro de modelos, después de guardar los modelos, versiones y etiquetas deseados, es posible rastrear el conjunto de datos, el modelo y el código específicos que se utilizaron para entrenar el modelo, así como el código específico que se utilizó para procesar los datos, cargar el tokenizador y el modelo, codificar, generar y descodificar el texto, guardar el modelo, finalizar el modelo, evaluar el modelo utilizando las métricas de perplexity `snapshot_id’s and your chosen metrics to MLflow by choosing the corerct experiment under `mlrun` del menú activo, las pestañas de NLTK y perplexity correspondientes:

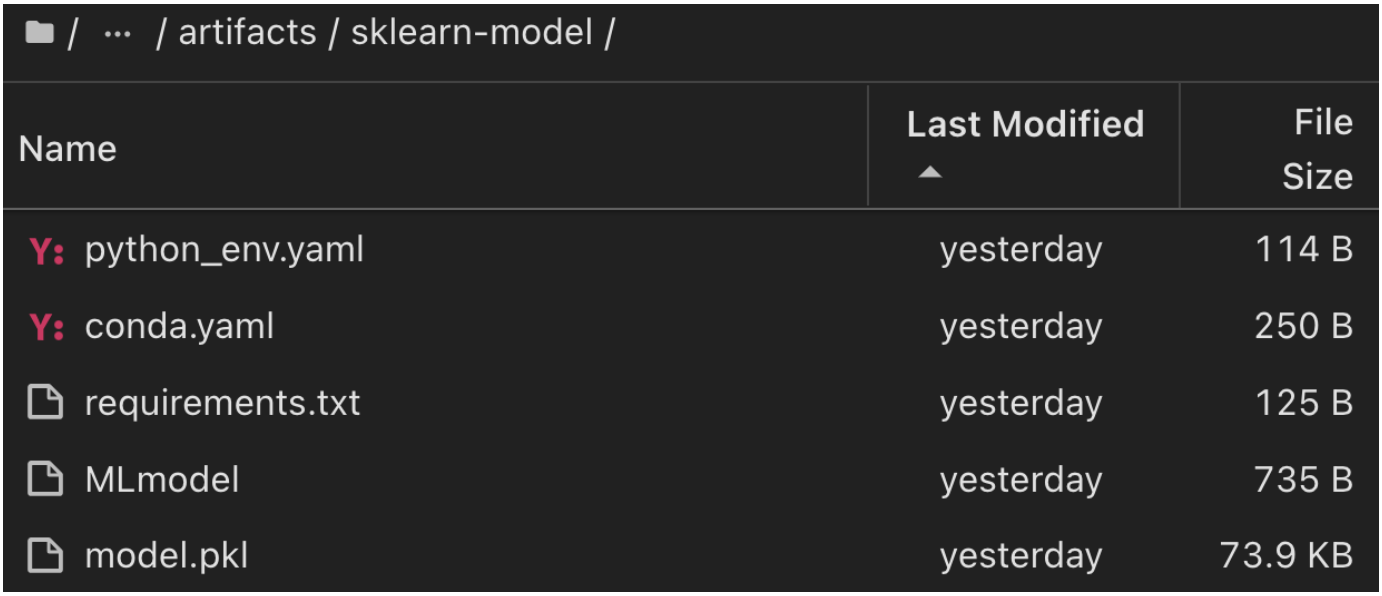
/ ··· / mlruns / 0 /

Name	Last Modified
d1b60feeef95498e9f9650f4717cfd00	yesterday
e594a0bd159e4a0385b70b2fcc245...	10 days ago
ffd362a7b84741238fea758f61ca933b	10 days ago
594a2ea3f02f474bb40d0483c10c4...	10 days ago
a819b9464f6c4825b1775d3bc89aa...	12 days ago
a109af0d7dc04d23b91f3fec93c939...	12 days ago
d956baa62bbf4fdf9a7b7692e85ca2...	12 days ago
859487a119c344dbbf4053674e9fd...	12 days ago
a13bd03ab91a4a7f847d953ce96f8f...	12 days ago
7e0df1fb96af499a83e55ef132ed86dd	12 days ago
21af6c1b1afe4b2fab4c7902ac6cfefc	12 days ago
da632683d23b48bd825196fdc7b72...	12 days ago
b5977cc431c14376a0b0daa5d72ab...	13 days ago
8bdb272a3837439da6e9d4cc72844...	13 days ago
18748118e867465c8974f553533d5...	13 days ago
db986997f61e4abe9a63c96c69973f...	13 days ago
7f67342558f54b2ca0687c321754f2...	13 days ago
0a52680a927d4368bc493f68c927e...	14 days ago
3d68fa51a58248538d32ca72be8cb...	14 days ago
f86685b664ed4f5199f5666519b41...	14 days ago
b8814d8376fc4c6aafa6c471496c0a7f	14 days ago
ab7c1e29302345cda9ad8c5a1635d...	14 days ago
652808daa03045198556af61d4d26...	14 days ago
e1d6df0c834c487bb103dd3553ab4...	14 days ago
7a28f96275814ce6bce1a2f17774c9...	14 days ago

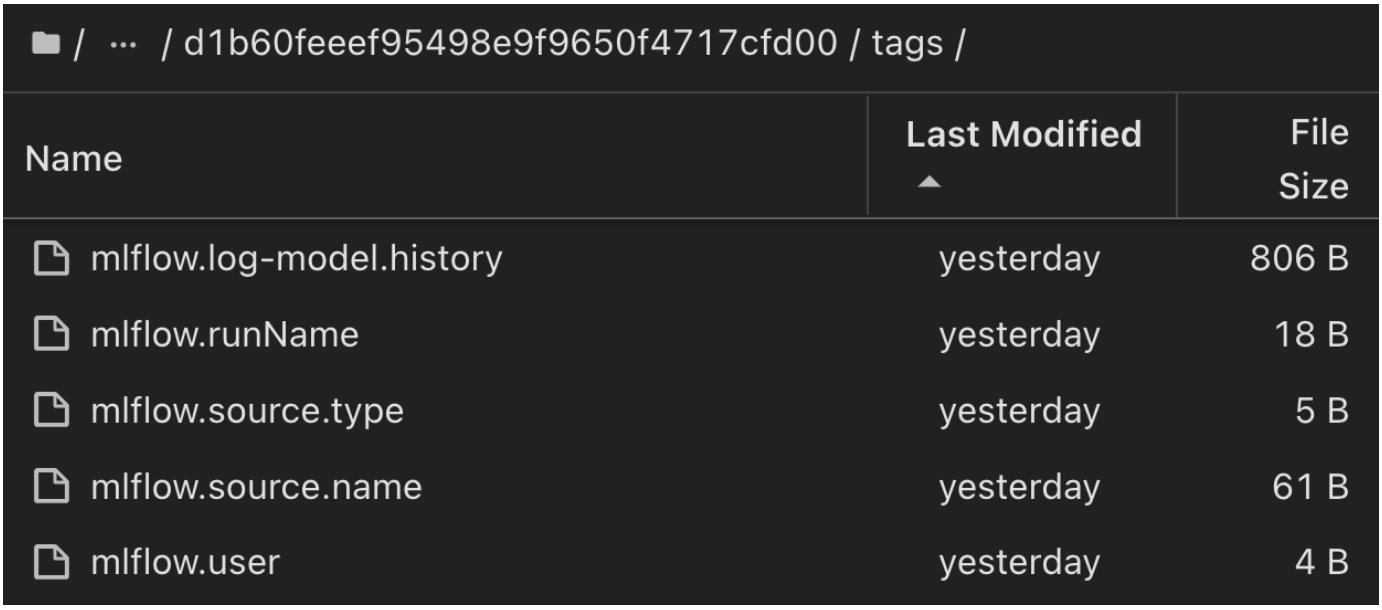
Del mismo modo, para nuestros `phi-2_finetuned_model` cuyos pesos cuantificados se calcularon a través de GPU o vGPU utilizando la `torch` biblioteca, podemos inspeccionar los siguientes artefactos intermedios, lo que permitiría la optimización del rendimiento, la escalabilidad (rendimiento/gaurantee SLA) y la reducción de costos de todo el flujo de trabajo:


/ ··· / 21af6c1b1afe4b2fab4c7902ac6cfefc / params /
Name
Last Modified
lr_scheduler_kwargs 12 days ago
lr_scheduler_type 12 days ago
optim_target_modules 12 days ago
max_steps 12 days ago
neftune_noise_alpha 12 days ago
num_train_epochs 12 days ago
include_num_input_tokens_seen 12 days ago
max_grad_norm 12 days ago
adam_epsilon 12 days ago
include_tokens_per_second 12 days ago
adam_beta2 12 days ago
split_batches 12 days ago
adam_beta1 12 days ago
dispatch_batches 12 days ago
torch_compile_mode 12 days ago
weight_decay 12 days ago
torch_compile_backend 12 days ago
learning_rate 12 days ago
torch_compile 12 days ago
torch_empty_cache_steps 12 days ago
ddp_timeout 12 days ago
eval_delay 12 days ago
eval_accumulation_steps 12 days ago
ray_scope 12 days ago
gradient_accumulation_steps 12 days ago
torchdynamo 12 days ago

Para una sola ejecución de experimento con Scikit-learn y MLflow, la siguiente figura muestra los artefactos generados, `conda` entorno, `MLmodel` archivo y `MLmodel` directorio:



Los clientes pueden especificar etiquetas, por ejemplo, «predeterminada», «etapa», «proceso» o «cuello de botella» para organizar distintas características de sus ejecuciones de flujo de trabajo de IA, tomar nota de sus últimos resultados o `contributors` establecer un seguimiento del progreso de los desarrolladores del equipo de ciencia de datos. Si para la etiqueta por defecto ' ', su guardado `mlflow.log-model.history`, , , , `mlflow.runName` `mlflow.source.type` `mlflow.source.name` y `mlflow.user` en la pestaña JupyterHub del navegador de archivos activo actualmente:



Por último, los usuarios tienen su propio Jupyter Workspace especificado, que se versiona y almacena en una reclamación de volumen persistente (PVC) en el clúster de Kubernetes. En la siguiente figura se muestra el espacio de trabajo Jupyter, que contiene el `netapp_dataops.k8s` paquete Python, y los resultados de una `VolumeSnapshot` :

Import NetApp DataOps Toolkit for Kubernetes functions

```
[1]: from netapp_dataops.k8s import list_volumes, list_volume_snapshots, create_volume_snapshot
```

Create Volume Snapshot for User Workspace Volume

The following example shows the execution of a "create volume snapshot" operation for my user workspace volume.

```
[2]: jupyterhub_namespace = "jupyterhub"
my_user_workspace_vol = "claim-moglesby"

create_volume_snapshot(namespace=jupyterhub_namespace, pvc_name=my_user_workspace_vol, print_output=True)
```

```
Creating VolumeSnapshot 'ntap-dsutil.20240726002955' for PersistentVolumeClaim (PVC) 'claim-moglesby' in namespace 'jupy
terhub'.
VolumeSnapshot 'ntap-dsutil.20240726002955' created. Waiting for Trident to create snapshot on backing storage.
Snapshot successfully created.
```

Nuestra tecnología probada en la industria Snapshot® y otras tecnologías se utilizaron para garantizar la protección de datos a nivel empresarial, el movimiento y la compresión eficiente. Para otros casos prácticos de IA, consulte "AIPod de NetApp"la documentación.

Información de copyright

Copyright © 2024 NetApp, Inc. Todos los derechos reservados. Imprimido en EE. UU. No se puede reproducir este documento protegido por copyright ni parte del mismo de ninguna forma ni por ningún medio (gráfico, electrónico o mecánico, incluidas fotocopias, grabaciones o almacenamiento en un sistema de recuperación electrónico) sin la autorización previa y por escrito del propietario del copyright.

El software derivado del material de NetApp con copyright está sujeto a la siguiente licencia y exención de responsabilidad:

ESTE SOFTWARE LO PROPORCIONA NETAPP «TAL CUAL» Y SIN NINGUNA GARANTÍA EXPRESA O IMPLÍCITA, INCLUYENDO, SIN LIMITAR, LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN CONCRETO, CUYA RESPONSABILIDAD QUEDA EXIMIDA POR EL PRESENTE DOCUMENTO. EN NINGÚN CASO NETAPP SERÁ RESPONSABLE DE NINGÚN DAÑO DIRECTO, INDIRECTO, ESPECIAL, EJEMPLAR O RESULTANTE (INCLUYENDO, ENTRE OTROS, LA OBTENCIÓN DE BIENES O SERVICIOS SUSTITUTIVOS, PÉRDIDA DE USO, DE DATOS O DE BENEFICIOS, O INTERRUPCIÓN DE LA ACTIVIDAD EMPRESARIAL) CUALQUIERA SEA EL MODO EN EL QUE SE PRODUJERON Y LA TEORÍA DE RESPONSABILIDAD QUE SE APLIQUE, YA SEA EN CONTRATO, RESPONSABILIDAD OBJETIVA O AGRAVIO (INCLUIDA LA NEGLIGENCIA U OTRO TIPO), QUE SURJAN DE ALGÚN MODO DEL USO DE ESTE SOFTWARE, INCLUSO SI HUBIEREN SIDO ADVERTIDOS DE LA POSIBILIDAD DE TALES DAÑOS.

NetApp se reserva el derecho de modificar cualquiera de los productos aquí descritos en cualquier momento y sin aviso previo. NetApp no asume ningún tipo de responsabilidad que surja del uso de los productos aquí descritos, excepto aquello expresamente acordado por escrito por parte de NetApp. El uso o adquisición de este producto no lleva implícita ninguna licencia con derechos de patente, de marcas comerciales o cualquier otro derecho de propiedad intelectual de NetApp.

Es posible que el producto que se describe en este manual esté protegido por una o más patentes de EE. UU., patentes extranjeras o solicitudes pendientes.

LEYENDA DE DERECHOS LIMITADOS: el uso, la copia o la divulgación por parte del gobierno están sujetos a las restricciones establecidas en el subpárrafo (b)(3) de los derechos de datos técnicos y productos no comerciales de DFARS 252.227-7013 (FEB de 2014) y FAR 52.227-19 (DIC de 2007).

Los datos aquí contenidos pertenecen a un producto comercial o servicio comercial (como se define en FAR 2.101) y son propiedad de NetApp, Inc. Todos los datos técnicos y el software informático de NetApp que se proporcionan en este Acuerdo tienen una naturaleza comercial y se han desarrollado exclusivamente con fondos privados. El Gobierno de EE. UU. tiene una licencia limitada, irrevocable, no exclusiva, no transferible, no sublicenciable y de alcance mundial para utilizar los Datos en relación con el contrato del Gobierno de los Estados Unidos bajo el cual se proporcionaron los Datos. Excepto que aquí se disponga lo contrario, los Datos no se pueden utilizar, desvelar, reproducir, modificar, interpretar o mostrar sin la previa aprobación por escrito de NetApp, Inc. Los derechos de licencia del Gobierno de los Estados Unidos de América y su Departamento de Defensa se limitan a los derechos identificados en la cláusula 252.227-7015(b) de la sección DFARS (FEB de 2014).

Información de la marca comercial

NETAPP, el logotipo de NETAPP y las marcas que constan en http://www.netapp.com/TM son marcas comerciales de NetApp, Inc. El resto de nombres de empresa y de producto pueden ser marcas comerciales de sus respectivos propietarios.