sklearn

Because the underlying model is scikit-learn, the cleanest way is: unpack the SageMaker TAR, `pickle`-load the sklearn estimator, then re-log it as an MLflow sklearn model to your Databricks tracking server.[1] [2]

Below is a minimal, end-to-end path tailored for sklearn.

1. One-time extract and load from S3 TAR

In any Python environment that has access to S3 (can be your laptop or a Databricks notebook):

```
import os, tarfile, tempfile, boto3, pickle

S3_BUCKET = "your-bucket"
S3_KEY = "path/to/model.tar.gz"

local_dir = tempfile.mkdtemp()
local_tar = os.path.join(local_dir, "model.tar.gz")

s3 = boto3.client("s3")
s3.download_file(S3_BUCKET, S3_KEY, local_tar)

with tarfile.open(local_tar, "r:gz") as tar:
    tar.extractall(local_dir)

print("Extracted to:", local_dir, os.listdir(local_dir))
```

Inside the extracted folder, you should see your sklearn model file (for SageMaker this is often something like `model.joblib`, `model.pkl`, or similar). If you saved it yourself in the training script with joblib/pickle, you already know the filename.[3]

Load the estimator:

```
model_path = os.path.join(local_dir, "model.pkl")  # or model.joblib etc.
with open(model_path, "rb") as f:
    sk_model = pickle.load(f)
```

At this point `sk_model` is a plain scikit-learn estimator with `.predict()` ready to use.[4]

2. Log it as an MLflow sklearn model to Databricks

Configure MLflow to talk to Databricks:

```
export MLFLOW_TRACKING_URI=databricks
export DATABRICKS_HOST="https://<your-dbx-workspace>"
export DATABRICKS_TOKEN="<personal-access-token>"
```

Then log the model:

```
import mlflow
import mlflow.sklearn
from mlflow.models import infer_signature
import pandas as pd

# Optional: sample data to define signature
X_sample = ...  # pandas DataFrame or numpy array used for training
y_pred_sample = sk_model.predict(X_sample)

mlflow.set_experiment("/Shared/sagemaker-imports")  # or UC experiment

with mlflow.start_run() as run:
    signature = infer_signature(X_sample, y_pred_sample)
    mlflow.sklearn.log_model(
        sk_model=sk_model,
        name="model",                 # artifact name in the run
        signature=signature,
        input_example=X_sample[:5],   # optional but useful
    )
    run_id = run.info.run_id
    print("MLflow run:", run_id)
```

`mlflow.sklearn.log_model` will create an MLflow model with both the sklearn flavor and the generic `pyfunc` flavor, which Databricks knows how to serve and use in batch.[5] [1]

If you prefer to keep the raw pickle as a separate artifact and wrap it, you can also use a small `pyfunc` wrapper as in the previous message, but for sklearn this direct flavor is simpler.[6] [1]

3. Register in Databricks Model Registry / Unity Catalog

Once the run has the logged model:

```
import mlflow

model_uri = f"runs:/{run_id}/model"  # 'model' = artifact name above

mv = mlflow.register_model(
    model_uri=model_uri,
    name="sagemaker_sklearn_import"  # or UC name: 'main.ml_models.sagemaker_sklearn_impo
)
```

```
print("Registered:", mv.name, "version:", mv.version)
```

You now have a proper MLflow model entry on Databricks that came from the SageMaker TAR without retraining.[7] [8]

4. Use from Databricks (serving or batch)

- For serving: create a Databricks Model Serving endpoint from this registered model (via UI or API) and call it from your Spring Boot services over HTTPS.
- For batch: in Databricks notebooks/Jobs, load it with `mlflow.pyfunc.load_model("models:/sagemaker_sklearn_import/Production")` once you transition a version to Production.[9] [7]

If you paste the extracted TAR directory listing (e.g., `ls -R` output) and, if relevant, how you saved the sklearn model in the training script, I can give you a ready-to-run script with the exact filenames wired in.

⁂

1. https://mlflow.org/docs/latest/python_api/mlflow.sklearn.html
2. https://mlflow.org/docs/3.1.3/ml/traditional-ml/sklearn/guide/
3. https://stackoverflow.com/questions/73042521/how-to-use-scikit-learn-model-from-inside-sagemaker-model-tar-gz-file
4. https://mlflow.org/docs/latest/ml/traditional-ml/sklearn/
5. https://github.com/mlflow/mlflow/blob/master/mlflow/sklearn/**init**.py
6. https://mlflow.org/docs/latest/python_api/mlflow.pyfunc.html
7. https://learn.microsoft.com/en-us/azure/databricks/mlflow/models
8. https://mlflow.org/docs/latest/ml/model-registry/
9. https://docs.databricks.com/aws/en/machine-learning/mlops/mlops-workflow
10. http://mlflow.org/blog/custom-pyfunc
11. https://mlflow.org/docs/latest/ml/traditional-ml/tutorials/creating-custom-pyfunc/part2-pyfunc-components/
12. https://docs.azure.cn/en-us/machine-learning/how-to-convert-custom-model-to-mlflow?view=azureml-api-2
13. https://www.marvelousmlops.io/p/logging-and-registering-models-with
14. https://stackoverflow.com/questions/67541920/how-to-create-a-model-on-model-tar-gz-in-sagemaker
15. https://github.com/amesar/mlflow-examples
16. https://github.com/mlflow/mlflow/issues/12775
17. https://www.youtube.com/watch?v=YWmnD_QcZQU
18. https://learn.microsoft.com/en-us/azure/machine-learning/how-to-log-mlflow-models?view=azureml-api-2