Lab 5: MLflow Model Registry

The MLflow Model Registry component is a centralized model store, set of APIs, and UI, to collaboratively manage the full lifecycle of an MLflow Model. It provides model lineage (which MLflow experiment and run produced the model), model versioning, model aliasing, model tagging, and annotations.

API Workflow

One method to interact with Model Registry is using the MLflow model flavor or MLflow Client Tracking API interface. In particular, you can register a model during an MLflow experiment run or after all your experiment runs.

Adding an MLflow Model to the Model Registry

There are three programmatic ways to add a model to the registry. First, you can use the `mlflow.<model_flavor>.log_model()` method. For example, in your code:

```
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

mlflow.set_tracking_uri("http://127.0.0.1:8081")

with mlflow.start_run() 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",
    )
```

In the above code snippet, if a registered model with the name doesn't exist, the method registers a new model and creates Version 1. If a registered model with the name exists, the method creates a new model version.

The second way is to use the mlflow.register_model() method, after all your experiment runs complete and when you have decided which model is most suitable to add to the registry. For this method, you will need the run_id as part of the runs:URI argument.

```
# Note: You can get the value from MLFLow UI

result = mlflow.register_model(
    "UPDATE_HERE", "sk-learn-random-forest-reg"
)
# Example:
# result = mlflow.register_model(
#     "runs:/1eb52d7d4a354856a52dd8598462d87d/sklearn-model", "sk-learn-random-forest-reg"
# )
```

If a registered model with the name doesn't exist, the method registers a new model, creates Version 1, and returns a ModelVersion MLflow object. If a registered model with the name exists, the method creates a new model version and returns the version object.

And finally, you can use the create_registered_model() to create a new registered model. If the model name exists, this method will throw an MlflowException because creating a new registered model requires a unique name.

```
from mlflow import MlflowClient

client = MlflowClient()
client.create_registered_model("sk-learn-random-forest-reg-model")

# Note: You will get and error since model "sk-learn-random-forest-reg-model" already exists.
```

The method above creates an empty registered model with no version associated. You can use create_model_version() as shown below to create a new version of the model.

```
client = MlflowClient()
result = client.create_model_version(
    name="sk-learn-random-forest-reg-model",
    source="UPDATE_HERE",
    run_id="UPDATE_HERE",
)
# Note: You can get the above values from MLFLow UI

## Example:
# result = client.create_model_version(
#     name="sk-learn-random-forest-reg-model",
#     source="mlflow-artifacts:/0/1eb52d7d4a354856a52dd8598462d87d/artifacts/sklearn-model",
#     run_id="1eb52d7d4a354856a52dd8598462d87d",
# )
```

Fetching an MLflow Model from the Model Registry

After you have registered an MLflow model, you can fetch that model using mlflow..load_model(), or more generally, load_model(). You can use the loaded model for one off predictions or in inference workloads such as batch inference.

Fetch a specific model version

To fetch a specific model version, just supply that version number as part of the model URI.

```
import mlflow.pyfunc

model_name = "sk-learn-random-forest-reg-model"
model_version = 1

model = mlflow.pyfunc.load_model(model_uri=f"models:/{model_name}/{model_version}")

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
)

model.predict(X_test)
```

Archiving an MLflow Model

You can move models versions out of a Production stage into an Archived stage. At a later point, if that archived model is not needed, you can delete it.

```
# Archive models version 1 from Production into Archived
client = MlflowClient()
client.transition_model_version_stage(
    name="sk-learn-random-forest-reg-model", version=1, stage="Archived"
)
```