Automatic Logging with MLflow Tracking

Auto logging is a powerful feature that allows you to log metrics, parameters, and models without the need for explicit log statements. All you need to do is to call mlflow.autolog() before your training code.

```
import mlflow

mlflow.autolog()

with mlflow.start_run():
    # your training code goes here
    ...
```

This will enable MLflow to automatically log various information about your run, including:

- **Metrics** - MLflow pre-selects a set of metrics to log, based on what model and library you use
- **Parameters** - hyper params specified for the training, plus default values provided by the library if not explicitly set
- **Model Signature** - logs Model signature instance, which describes input and output schema of the model
- **Artifacts** - e.g. model checkpoints
- **Dataset** - dataset object used for training (if applicable), such as tensorflow.data.Dataset

Lab Solution

Complete solution for this lab is available in the `lab6_autolog.ipynb` notebook.

How to Get started

Step 1 - Get MLflow

Make sure MLFlow is installed.

Step 2 - Insert `mlflow.autolog` in Your Code

For example, following code snippet shows how to enable autologging for a scikit-learn model:

```
import mlflow

from sklearn.model_selection import train_test_split
from sklearn.datasets import load_diabetes
from sklearn.ensemble import RandomForestRegressor

mlflow.autolog()

db = load_diabetes()
X_train, X_test, y_train, y_test = train_test_split(db.data, db.target)

rf = RandomForestRegressor(n_estimators=100, max_depth=6, max_features=3)
# MLflow triggers logging automatically upon model fitting
rf.fit(X_train, y_train)
```

Step 3 - Execute Your Code

Execute your code in the jupyter or python notebook.

Step 4 - View Your Results in the MLflow UI

Once your training job finishes, you can run following command to launch the MLflow UI:

```
mlflow ui --port 8081
```

Note: Run above command from same path as your notebook.

Then, navigate to http://localhost:8081 in your browser to view the results.

Customize Autologging Behavior

You can also control the behavior of autologging by passing arguments to `mlflow.autolog()` function. For example, you can disable logging of model checkpoints and assosiate tags with your run as follows:

```
import mlflow

mlflow.autolog(
    log_model_signatures=False,
    extra_tags={"YOUR_TAG": "VALUE"},
)
```

Enable / Disable Autologging for Specific Libraries

One common use case is to enable/disable autologging for a specific library. For example, if you train your model on tensorflow but use scikit-learn for data preprocessing, you may want to disable autologging for scikit-learn while keeping it enabled for tensorflow. You can achieve this by either (1) enable autologging only for tensorflow using tensorflow flavor (2) disable autologging for scikit-learn using its flavor with `disable=True`.

```
import mlflow

# Option 1: Enable autologging only for tensorflow
mlflow.tensorflow.autolog()

# Option 2: Disable autologging for scikit-learn, but enable it for other libraries
mlflow.sklearn.autolog(disable=True)
mlflow.autolog()
```