Published in Towards Data Science

Ram Vegiraju Follow
Apr 5 · 5 min read · · Listen

MLflow Model Serving

Hosting Models Locally With MLflow

Image from Unsplash by Matt Botsford

In a previous article we discussed how you can track and register models with MLflow.

Using MLflow models we can package our ML models for **local real-time inference** or batch inference on Apache Spark. For this article we'll explore how we can train a Sklearn model and then locally deploy it for inference using MLflow. We'll be using the following example from the MLflow repository as a reference. The code for this example in specific can be found here.

Table of Contents

1. Setup
2. Model Training
3. Deployment & Inference
4. Additional Resources & Conclusion

1. Setup

The setup is pretty simple here as MLflow is an open source package that you can work with.

```
pip install mlflow
```

mlflow.py hosted with ❤ by GitHub view raw

Install MLflow

If you want to get full access to the MLflow documentation, get started with their repository which also comes with a large set of examples covering all of the main components that we discussed.

Now for our setup for MLflow models we need to **define a few files in a certain structure that the model serving will understand.** We will capture our model that we will be deploying in a script, for this we can create a **train.py** where we do model training and **log** our **model artifacts.**

```
from sklearn.metrics import mean_squared_error
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn import metrics

import mlflow
import mlflow.sklearn
```

train.py hosted with ❤ by GitHub view raw

Script for Model Serving

We can point to this script in a MLproject file in the root directory. This file is where we can essentially provide the **metadata** necessary for **model serving**. Here we can capture our entry point script in train.py, this will essentially let MLflow know this is where we can capture our model artifacts in our project.

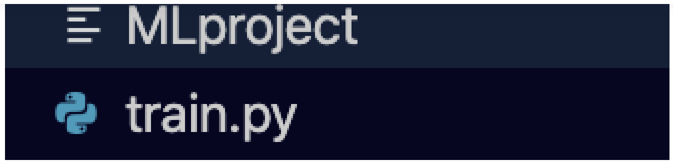
```
name: sklearn_regression_example

conda_env: conda.yaml

entry_points:
  main:
    command: "python train.py"
```

MLproject hosted with ❤ by GitHub view raw

MLProject defined

You can define further parameters in a yaml format in this file. The essence of an MLflow Project is to package your data science code in a reusable manner for deployment. In the yaml file above for example you can also define a conda environment if you are deploying in one.

We will not be working with a conda environment in this example, but you can specify your dependencies in this file. I've also attached a sample conda.yaml file just for reference in the code repository for this example. Your file structure should look like the following before we can get to working on our training script.



File Structure (Screenshot by Author)

2. Model Training

Before we can get to deploying our model, we'll train a simple Linear Regression model with the Boston Housing dataset using the Sklearn framework.

```python
if __name__ == "__main__":
  #Load data
  boston = datasets.load_boston()
  df = pd.DataFrame(boston.data, columns = boston.feature_names)
  df['MEDV'] = boston.target

  #Split Model
  X = df.drop(['MEDV'], axis = 1)
  y = df['MEDV']
  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .2, random_state = 42)

  #Model Creation
  lm = LinearRegression()
  lm.fit(X_train,y_train)

  #Model prediction
  Y_Pred = lm.predict(X_test)
  RMSE = np.sqrt(metrics.mean_squared_error(y_test, Y_Pred))
```

train.py hosted with ❤ by GitHub view raw

Model Training

We can capture our RMSE metric to track across different iterations of training with the log_metric API call that MLflow provides.

```python
print(f"RMSE: {RMSE}")
```

More importantly we need to capture our model artifacts/data for serving which we can do with a log_model API call. Even more nicely for sklearn, MLflow provides a specific sklearn.log_model call tailored for the framework.

```
mlflow.sklearn.log_model(lm, "model")
print("Model saved in run %s" % mlflow.active_run().info.run_uuid)
```

train.py hosted with ❤ by GitHub view raw

Capture Model Artifacts

Now if we run the training script we should see a run ID omitted from the last line. We capture the run ID in the last line of the previous gist so we can specify which model we want to deploy.

```
ramvegiraju@Rams-MacBook-Pro mlflow-serving % python3 train.py
RMSE: 4.928602182665329
Model saved in run a0ee696856d44bc0b231ee5dbe885847
```

Model Training + Run ID(screenshot by Author)

Make sure to save this run ID we will need it for our model serving.

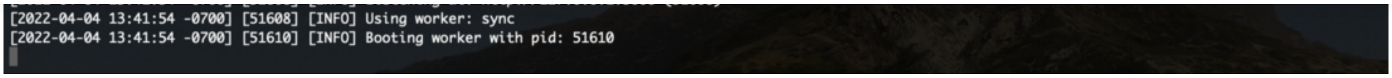
3. Deployment & Inference

Using this run ID we can serve our MLflow model as a public REST API. Use the following command and our model will be available for inference as a REST API.

```
mlflow models serve --model-uri runs:/caf16f1fe3c949ae9ba534367fd14c17/model --no-conda
```

serve.sh hosted with ❤ by GitHub view raw

Note we provide an **environment variable** for **no conda environment** as we do not have one for this case, but make sure to omit that if you are in a conda environment.



Model Serving (Screenshot by Author)

Now we can invoke the local REST API endpoint with a POST input to the **/invocations** path. We can grab a sample data point and specify our input data point as well as the content type (JSON, CSV).

```
curl -d '{"data":[[0.09178,0.0,4.05,0.0,0.51,6.416,84.1,2.6463,5.0,296.0,16.6,395.5,9.04]]}'
-H 'Content-Type: application/json'  localhost:5000/invocations
```

invoke.sh hosted with ❤ by GitHub view raw

Invoke

You can further add more customization to define your inputs and outputs through model signatures. You can add a model signature to the log_model call to either automatically infer the input data based off of the training data or specify your column/input data features.

For this example we'll just directly invoke the REST API with a sample data point as seen in the above shell command. Run the command in another shell and you should see inference.

Inference (Screenshot by Author)

4. Additional Resources & Conclusion

GitHub - RamVegiraju/mlflow-serving: Example of locally serving a MLflow Model

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or...

github.com

flexibility for you to choose which step you want to focus on in your ML project.

I hope this article was a good introduction to hosting models with MLflow, we'll explore different facets of the platform in coming articles.

Additional Resources

Tracking ML Experiments With MLflow

MLflow Model Registry & Serving

Deploying ML Models

If you enjoyed this article feel free to connect with me on LinkedIn and subscribe to my Medium Newsletter. If you're new to Medium, sign up using my Membership Referral.