MLflow documentation

MLflow is a platform to streamline machine learning development, including tracking experiments, packaging code into reproducible runs, and sharing and deploying models. MLflow offers a set of lightweight APIs that can be used with any existing machine learning application or library (TensorFlow, PyTorch, XGBoost, etc.), wherever you currently run ML code (e.g. in notebooks, standalone applications or the cloud). MLflow's current components are:

- **MLflow Tracking**: An API to log parameters, code, and results in machine learning experiments and compare them using an interactive UI.
- **MLflow Projects**: A code packaging format for reproducible runs using Conda and Docker, so you can share your ML code with others.
- **MLflow Models**: A model packaging format and tools that let you easily deploy the same model (from any ML library) to batch and real-time scoring on platforms such as Docker, Apache Spark, Azure ML and AWS Sage Maker.
- **MLflow Model Registry**: A centralized model store, set of APIs, and UI, to collaboratively manage the full lifecycle of MLflow Models.

Installing:

Install MLflow from PyPI via pip install mlflow
MLflow requires conda to be on the PATH for the projects feature.
Install a lower dependency subset of MLflow from PyPI via pip install mlflow-skinny Extra dependencies can be added per desired scenario. For example, pip install mlflow-skinny pandas NumPy allows for mlflow.pyfunc.log_model support.

open mlflow dashboard:-

```
mlflow ui --backend-store-uri sqlite:///mlflow.db
```

How to implement MLFlow in our application—

Step 1-Importing lib

```
import mlflow
```

Step 2 - Set the tracker and experiment

```
mlflow.set_tracking_uri(DATABASE_URI)
mlflow.set_experiment("EXPERIMENT_NAME")
```

Step 3 - Start a experiment run with

```
mlflow.start_run():
```

Step 4 - Logging the metadata

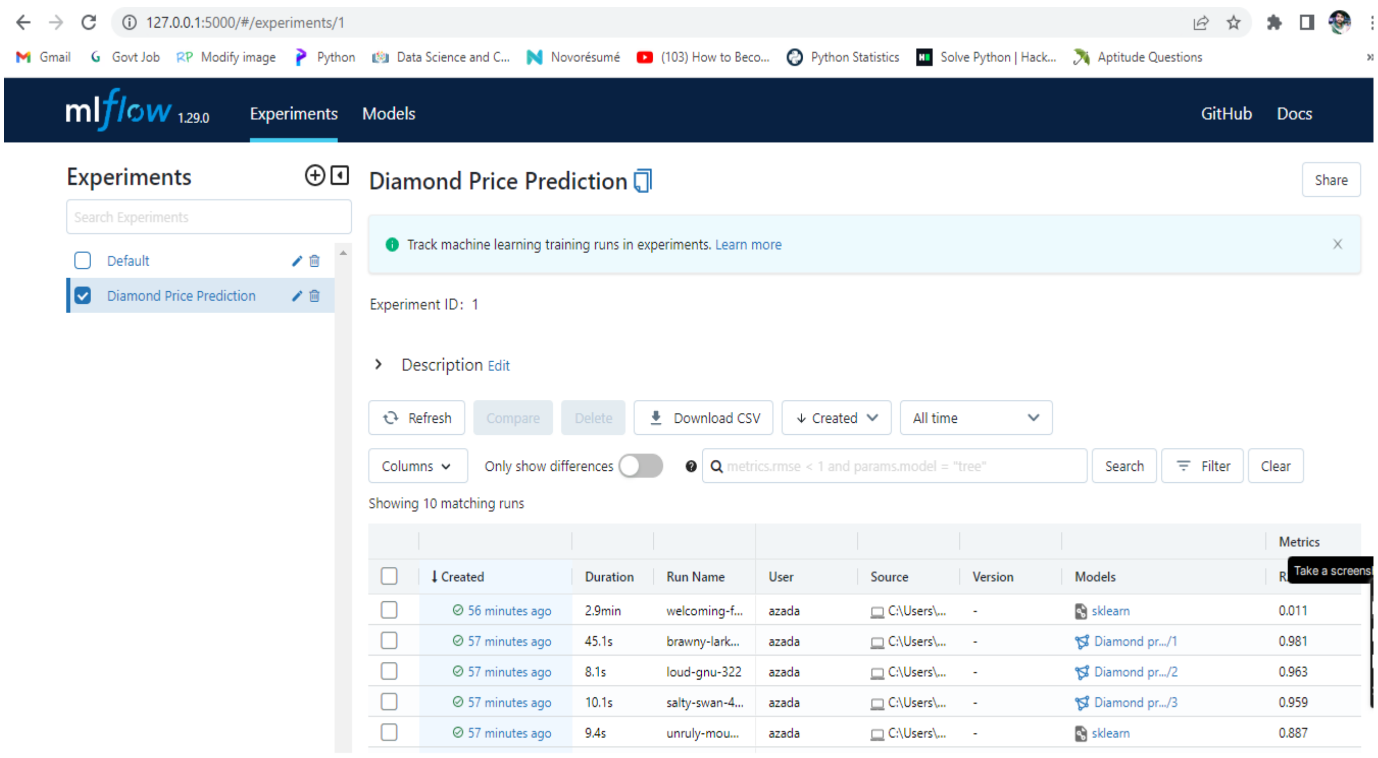
```
mlflow.set_tag(KEY, VALUE) mlflow.log_param(KEY, VALUE)
mlflow.log_metric(KEY, VALUE)
```

Step 5 - Logging the model and other files (2 ways)

Way 1 - mlflow..log_model(MODEL_OBJECT, artifact_path="PATH")

Way 2 - mlflow.log_artifact(LOCAL_PATH, artifact_path="PATH")

MLFlow Interface Experiment Tracking:



127.0.0.1:5000/#/experiments/1

mlflow 1.29.0 | Experiments | Models | GitHub | Docs

Experiments

Default

Diamond Price Prediction

Diamond Price Prediction

Track machine learning training runs in experiments. Learn more

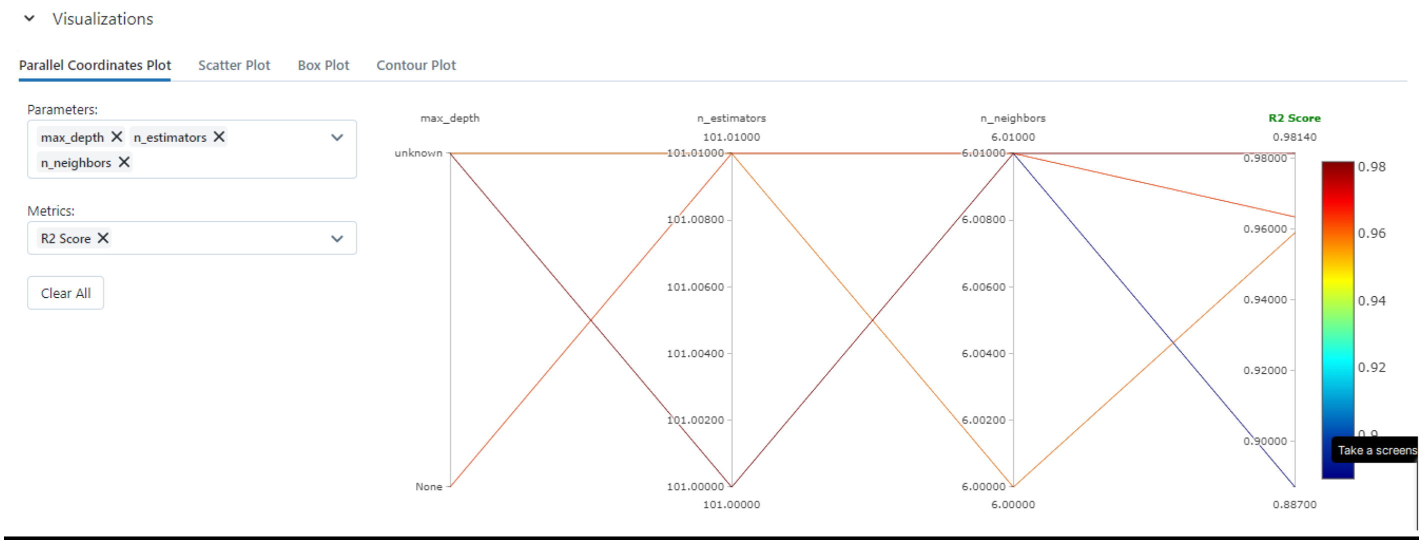
Experiment ID: 1

Description Edit

Showing 10 matching runs

Created	Duration	Run Name	User	Source	Version	Models	R
56 minutes ago	2.9min	welcoming-f...	azada	C:\Users\...	-	sklearn	0.011
57 minutes ago	45.1s	brawny-lark...	azada	C:\Users\...	-	Diamond pr.../1	0.981
57 minutes ago	8.1s	loud-gnu-322	azada	C:\Users\...	-	Diamond pr.../2	0.963
57 minutes ago	10.1s	salty-swan-4...	azada	C:\Users\...	-	Diamond pr.../3	0.959
57 minutes ago	9.4s	unruly-mou...	azada	C:\Users\...	-	sklearn	0.887

Parallel coordinates plot:



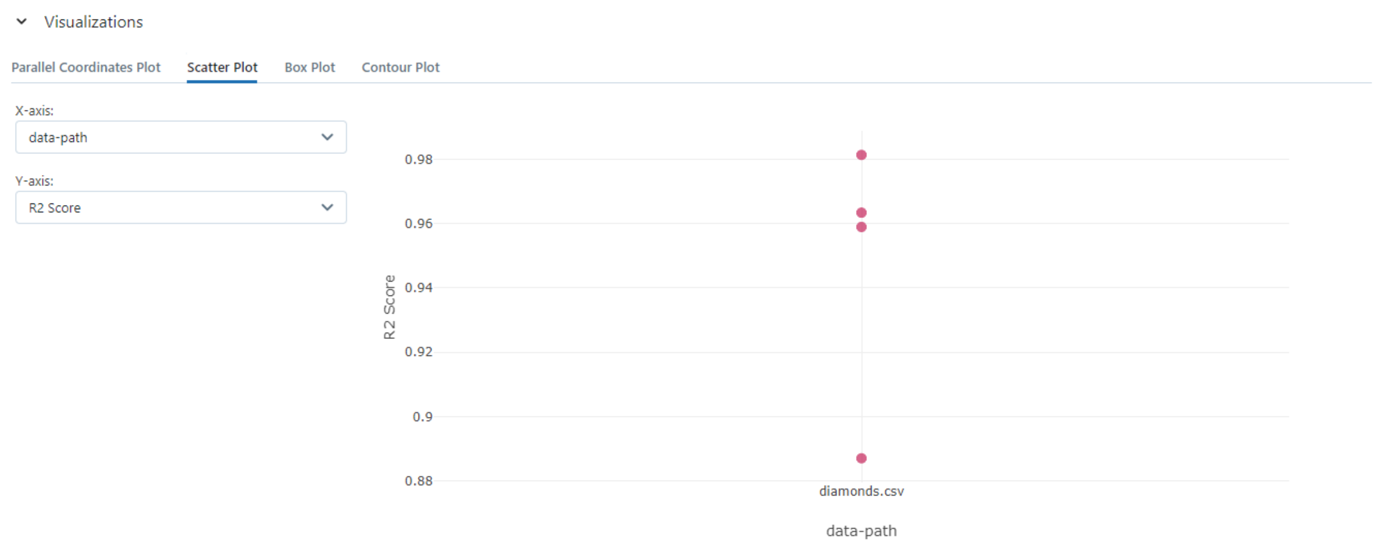
Visualizations

Parallel Coordinates Plot | Scatter Plot | Box Plot | Contour Plot

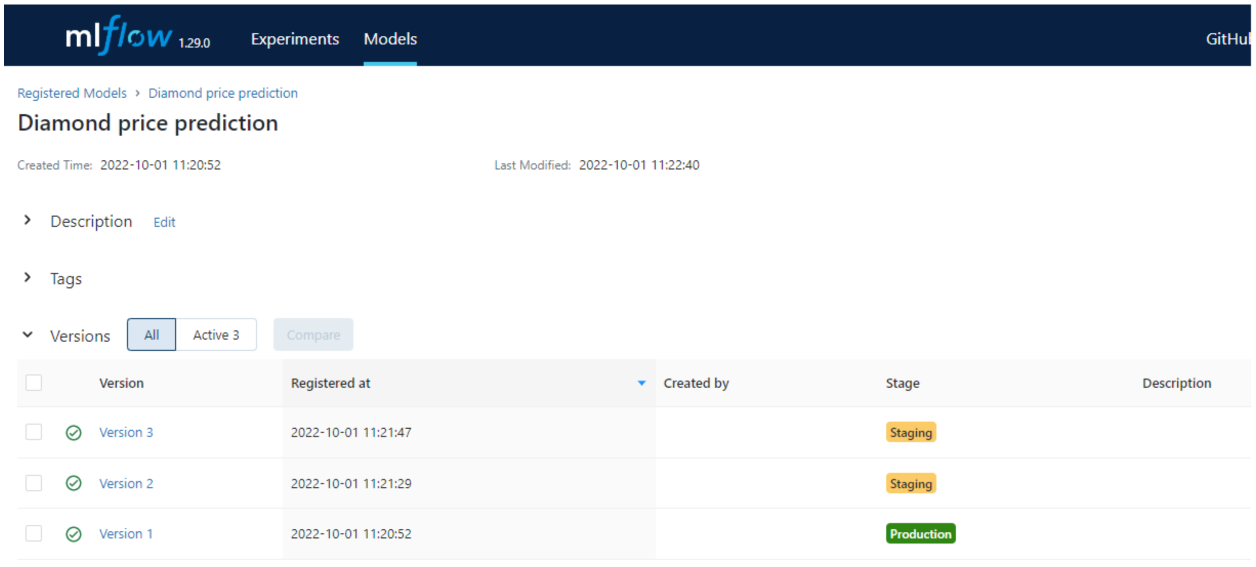
Parameters: max_depth, n_estimators, n_neighbors

Metrics: R2 Score

Scatter Plot:



MLFlow Model Management:



Diamond price prediction

Created Time: 2022-10-01 11:20:52   Last Modified: 2022-10-01 11:22:40

Version	Registered at	Created by	Stage	Description
Version 3	2022-10-01 11:21:47		Staging	
Version 2	2022-10-01 11:21:29		Staging	
Version 1	2022-10-01 11:20:52		Production	

Pipeline using Workflow Orchestration(Prefect)

What is Prefect?

Prefect is an open-sourced framework to build workflows in Python. Prefect makes it easy to build, run, and monitor data pipelines at scale.

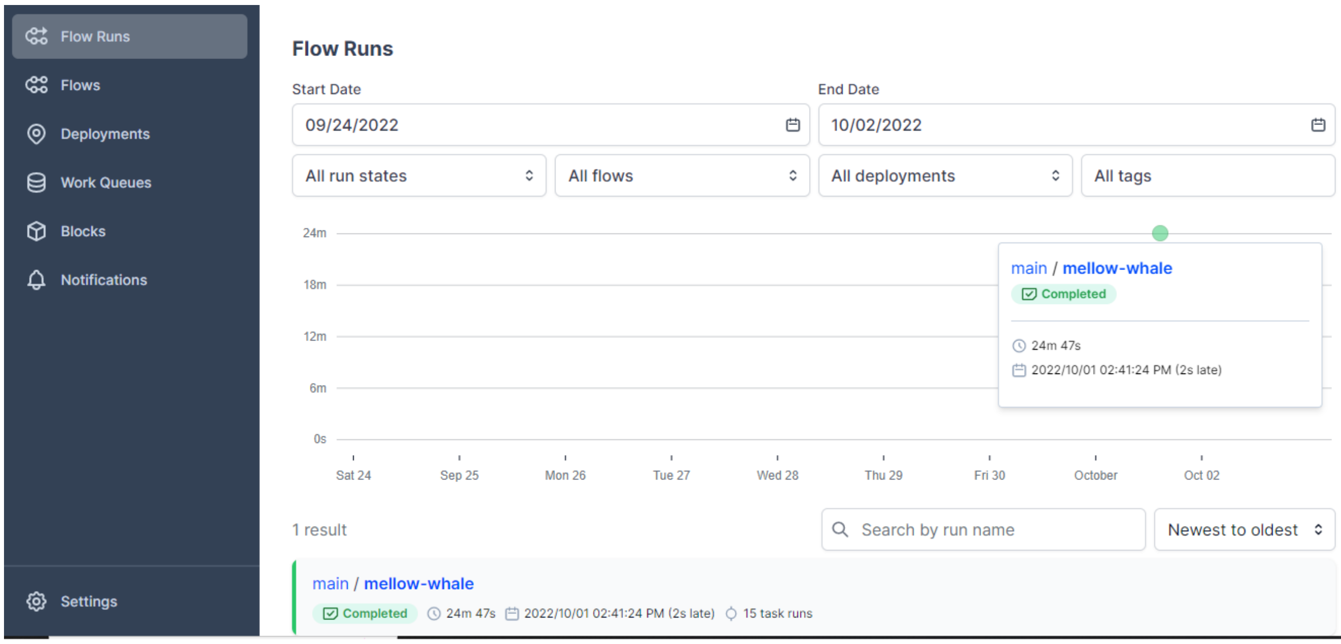
To install Prefect:

pip install prefect

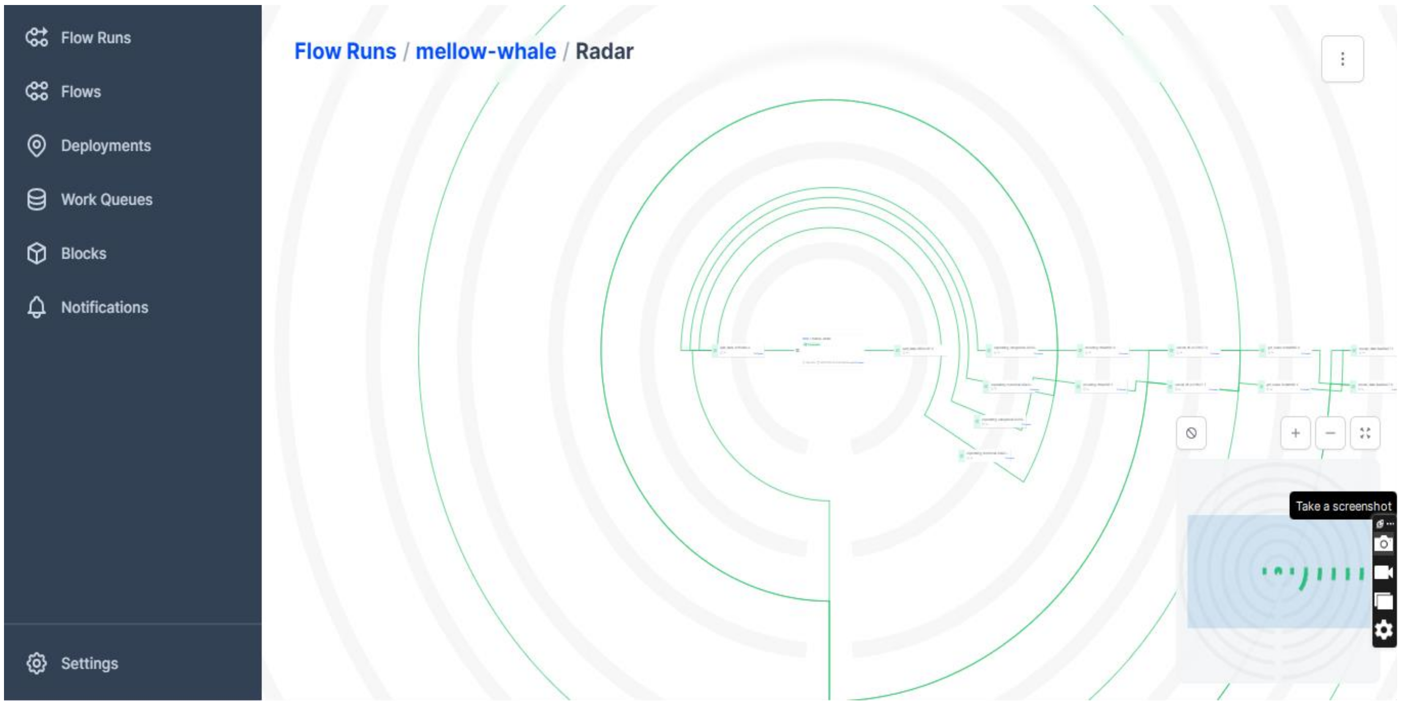
To open the prefect orion:

prefect orion start

Prefect orion Interface:



Radar:



****Thank you****