MLflow

Introduction to Experiment Tracking and Model Management

MLflow is an open-source platform to manage the ML lifecycle, including experimentation, reproducibility, deployment, and a central model registry.

MLflow currently offers four components:

- MLflow Tracking-Record and query experiments: code, data, config, and results
- MLflow projects-Package data science code in a format to reproduce runs on any platform
- MLflow models- Deploy Machine learning models in diverse serving environments
- Model Registry- Store, annotate, discover, and manage models in a central repository

MLflow: Tracking of Experiments, logging or recording all the Experiments.

Login is important for Organization of Production pipeline properly.

MLflow interface is helping to track for-

- What kind of Algorithm
- What kind of Hyper parameters
- What kind of scores we get for the model.

MLflow is an open source platform for managing the end-to-end machine learning lifecycle. It tackles four primary functions:

- Tracking experiments to record and compare parameters and results (MLflow Tracking).
- Packaging ML code in a reusable, reproducible form in order to share with other data scientists or transfer to production (MLflow Projects).
- Managing and deploying models from a variety of ML libraries to a variety of model serving and inference platforms (MLflow Models).
- Providing a central model store to collaboratively manage the full lifecycle of an MLflow Model, including model versioning, stage transitions, and annotations (MLflow Model Registry).

Concepts

MLflow Tracking is organized around the concept of *runs*, which are executions of some piece of data science code. Each run records the following information:

Code Version:

Git commit hash used for the run, if it was run from an MLflow Project.

Start & End Time

Start and end time of the run

Source

Name of the file to launch the run, or the project name and entry point for the run if run from an MLflow Project.

Parameters

Key-value input parameters of your choice. Both keys and values are strings.

Metrics

Key-value metrics, where the value is numeric. Each metric can be updated throughout the course of the run (for example, to track how your model's loss function is converging), and MLflow records and lets you visualize the metric's full history.

Artifacts

Output files in any format. For example, you can record images (for example, PNGs), models (for example, a pickled scikit-learn model), and data files (for example, a Parquet file) as artifacts.

Introduction to Experiment Tracking

Terminologies:

1. Experiment
2. Run
3. Metadata (i.e. Tags, Parameters, Metrics)
4. Artifacts (i.e. Output files associated with experiment runs)

What do you want to track for each Experiment Run?

1. Training and Validation Data Used
2. Hyperparameters
3. Metrics
4. Models

Why Track?

Organization Optimization Reproducibility

Tool - MLFlow

MLFlow helps you to organize your experiments into runs.

MLFlow keeps track of:

- Tags
- Parameters
- Metrics
- Models
- Artifact
- Source code, Start and End Time, Authors etc..

Run below mentioned commands to install mlflow on your system:

- pip install mlflow
- mlflow ui --backend-store-uri sqlite:///mlflow.db

Introduction to MLFlow

Step 1 - Import MLFlow

```
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)

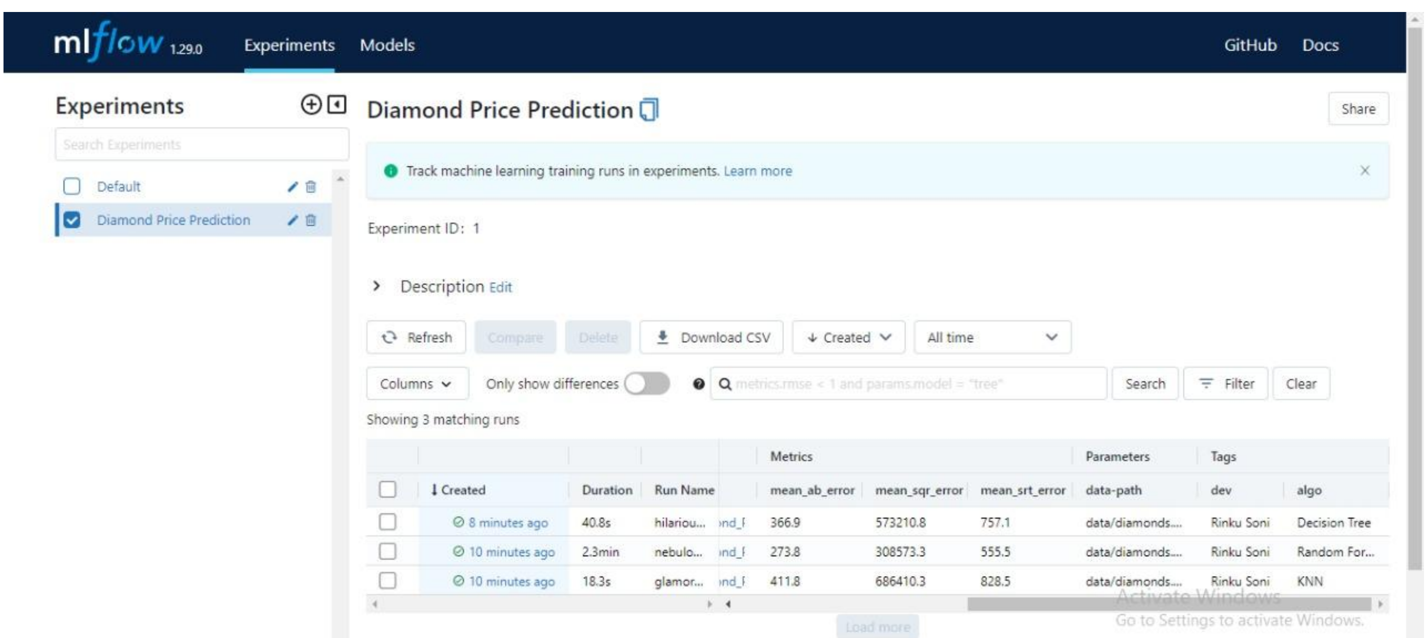
mlflow.log_param(KEY, VALUE) mlflow.log_metric(KEY, VALUE)

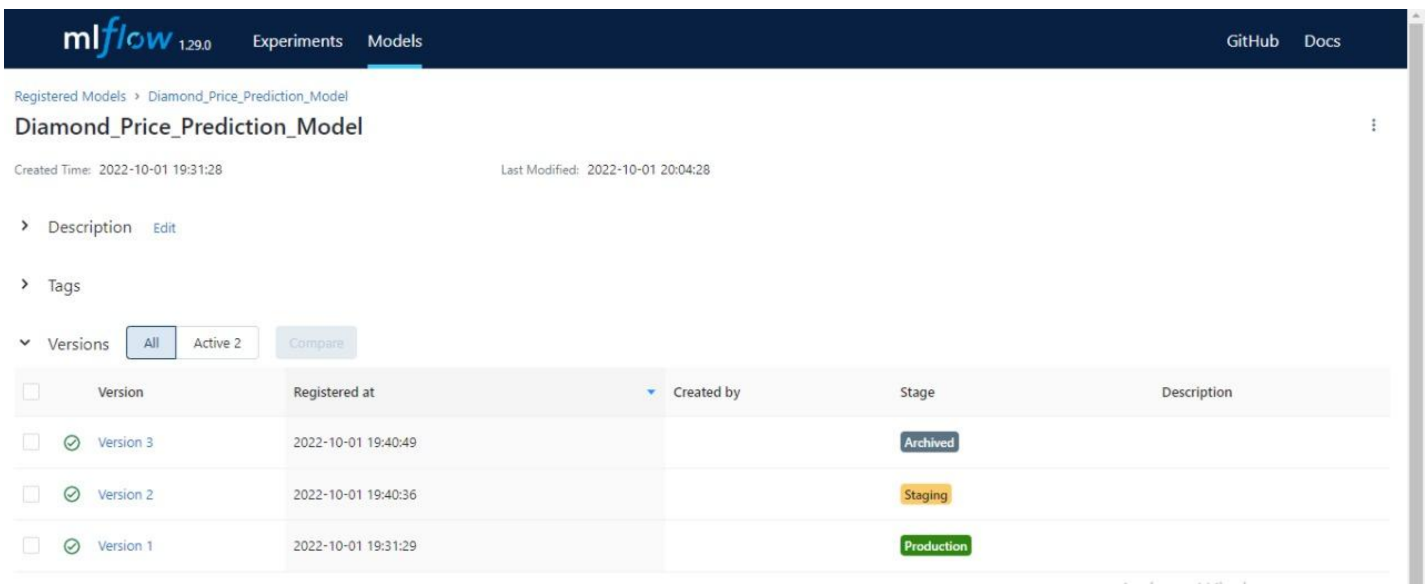
Step 5 - Logging the model and other files (2 ways)

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

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

Below are screenshots of Experiment tracking and Model Management of diamond price prediction models:





PREFECT ORCHESTRATE ML PIPELINES

Managing Machine Learning Workflows using the tool Prefect 2.0

Why Prefect?

- Python-based open source tool
- Manage ML Pipelines
- Schedule and Monitor the flow
- Gives observability into failures
- Native dask integration for scaling (Dask is used for parallel computing)

Creating and activating a Virtual Environment

In order to install prefect, create a virtual environment:

`$ python -m venv mlops`

Enter the Virtual Environment using below mentioned command:

`$ .\mlops\Scripts\activate`

Installing Prefect 2.0

Now install Prefect:

`$ pip install prefect`

OR if you have Prefect 1, upgrade to Prefect 2 using this command:

`$ pip install -U prefect`

OR to install a specific version:

`$ pip install prefect==2.4`

Check Prefect Version

Check the prefect version:

`$ prefect version`

Running Prefect Dashboard

`$ prefect orion start`

```
 ___ ___ ___ ___ ___ ___ _____     ___  ___ ___ ___  _  _
| _ \ _ \ __| __| __/ __|_   _|   / _ \| _ \_ _/ _ \| \| |
|  _/   / _|| _|| _| (__  | |    | (_) |   /| | (_) | .` |
|_| |_|_\___|_| |___\___| |_|     \___/|_|_\___\___/|_|\_|
Configure Prefect to communicate with the server with:
    prefect config set PREFECT_API_URL=http://127.0.0.1:4200/api
View the API reference documentation at http://127.0.0.1:4200/docs
Check out the dashboard at http://127.0.0.1:4200/
```

Note - In Windows OS, if your path contains spaces, it will generate error (as mentioned below) when you try to run prefect orion.

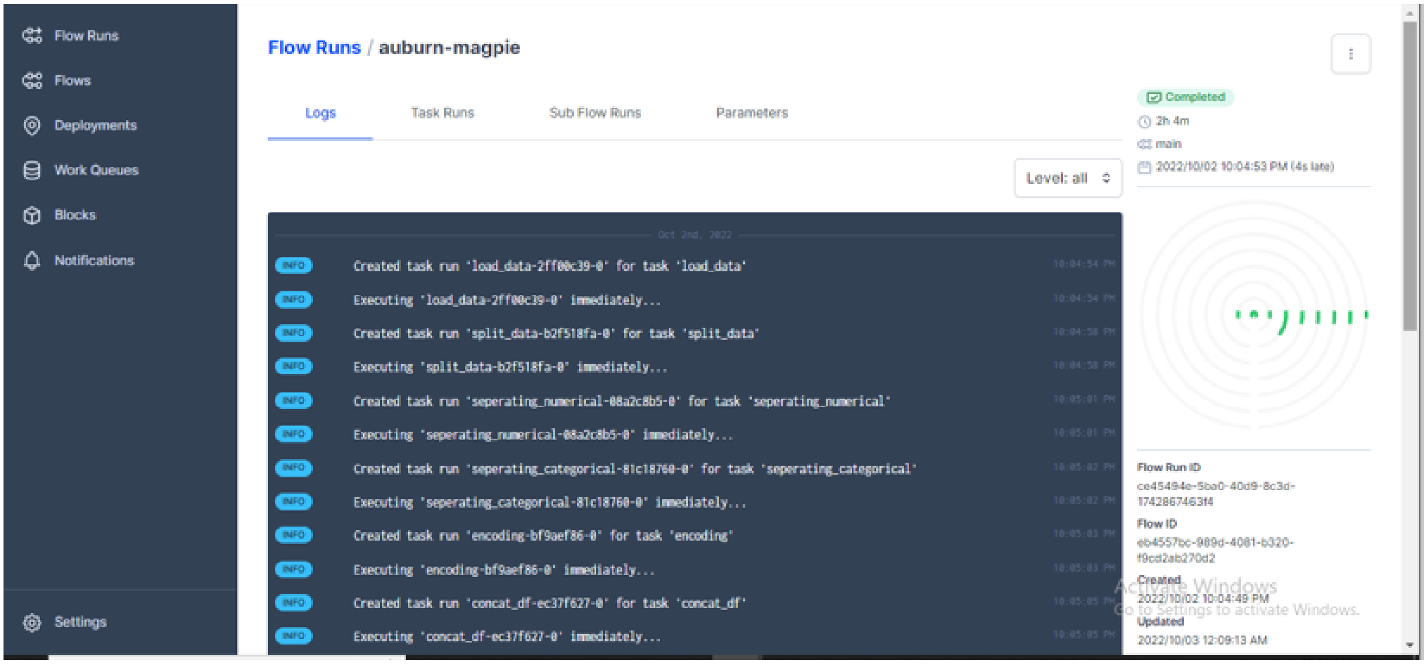
```
 ___ ___ ___ ___ ___ ___ _____    ___  ___ ___ ___  _  _
| _ \ _ \ __| __| __/ __|_   _|  / _ \| _ \_ _/ _ \| \| |
|  _/   / _|| _|| _| (__  | |   | (_) |   /| | (_) | .` |
|_| |_|_\___|_| |___\___| |_|    \___/|_|_\___\___/|_|\_|
Configure Prefect to communicate with the server with:
    prefect config set PREFECT_API_URL=http://127.0.0.1:4200/api
View the API reference documentation at http://127.0.0.1:4200/docs
Check out the dashboard at http://127.0.0.1:4200/
Usage: uvicorn [OPTIONS] APP

Try 'uvicorn --help' for help.

Error: Got unexpected extra argument (prefect.orion.api.server:create_app)
Orion stopped!
```

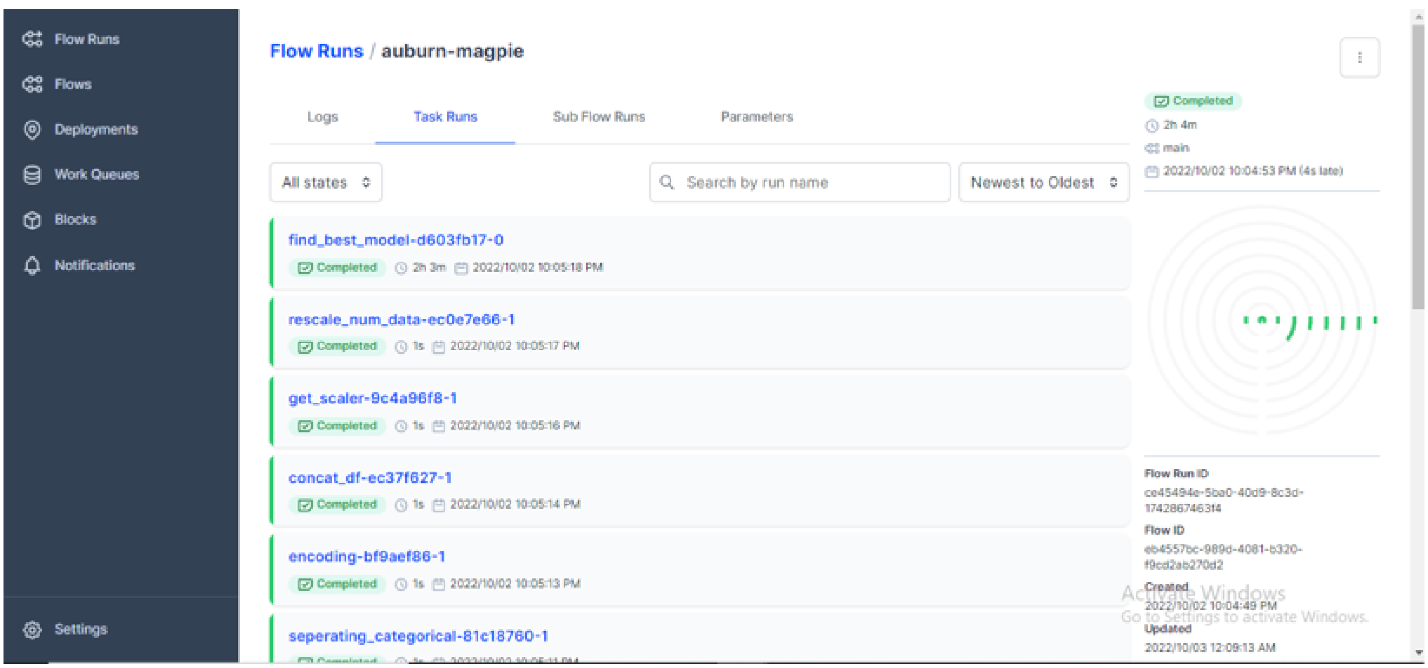
Deployment of Prefect Flow

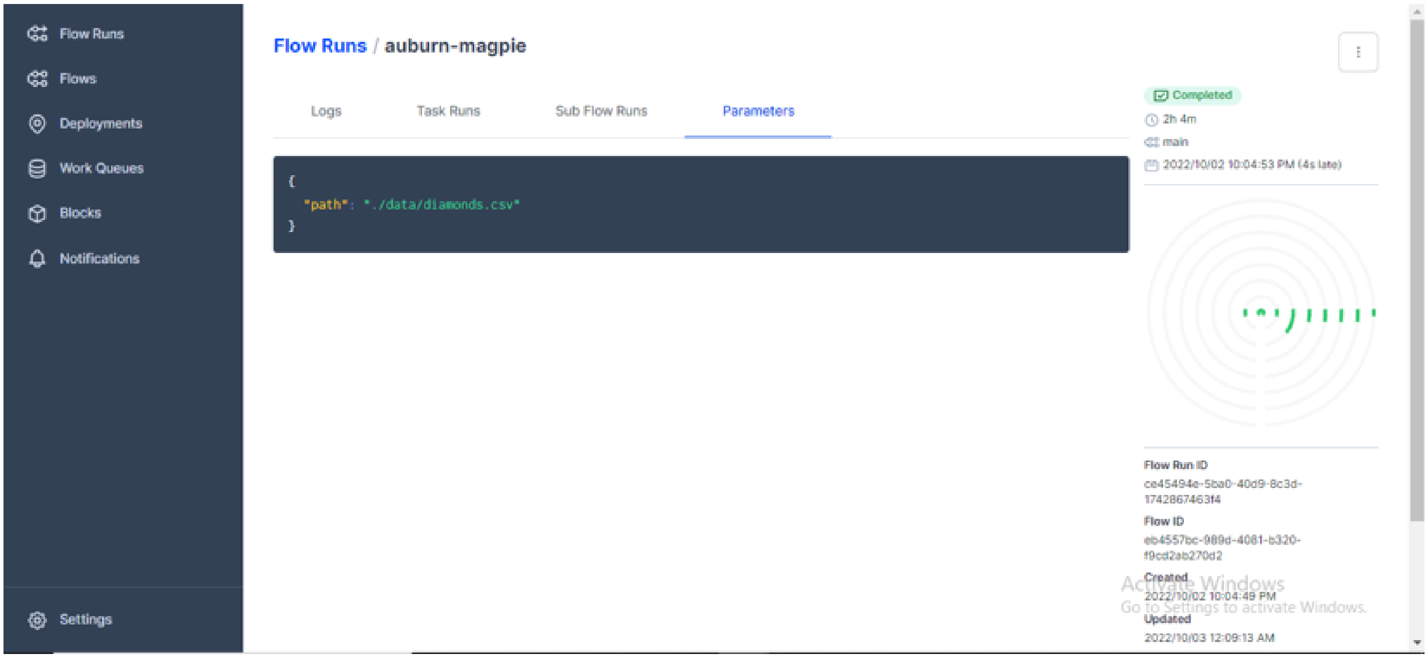
- work_queue_name is used to submit the deployment to a specific work queue.
- You don't need to create a work queue before using the work queue. A work queue will be created if it doesn't exist.

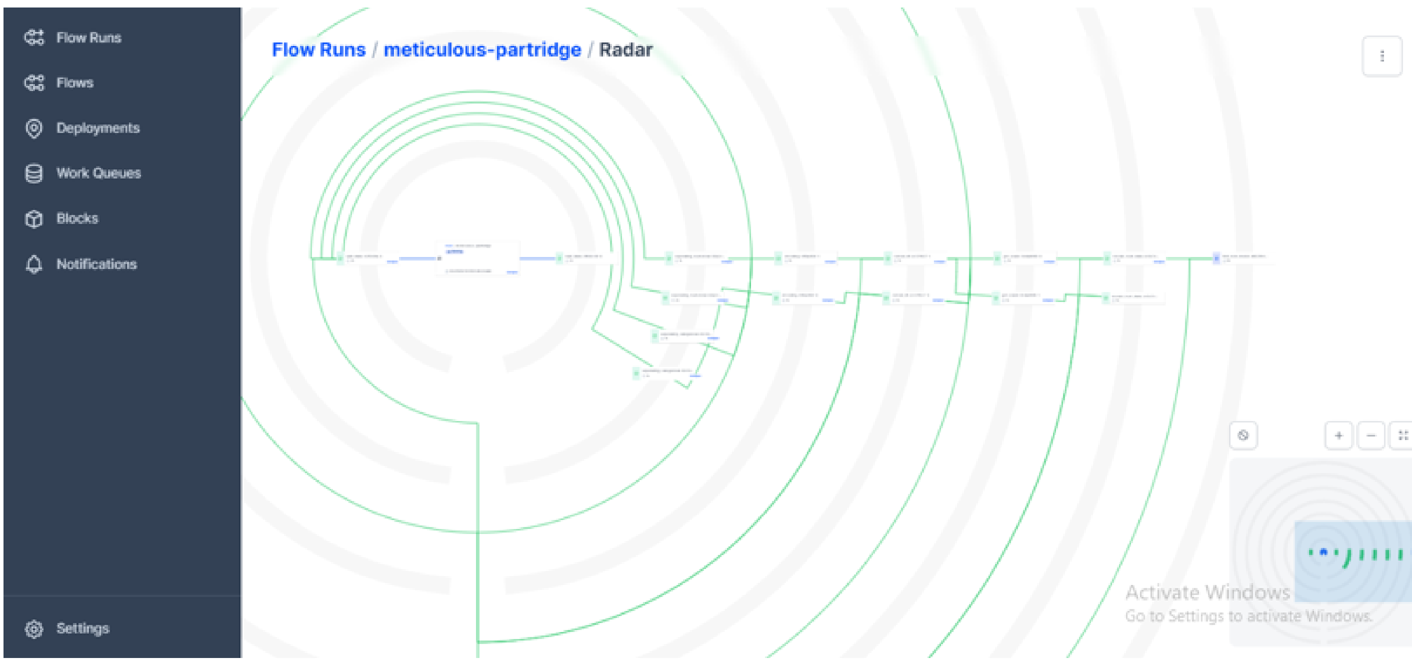
```
# Deploy the main function
from prefect.deployments import Deployment
from prefect.orion.schemas.schedules import IntervalSchedule
from datetime import timedelta

deployment = Deployment.build_from_flow(
    flow=main,
    name="Diamond_Price_Prediction_Training",
    schedule=IntervalSchedule(interval=timedelta(days=7)),
    work_queue_name="ml"
)

deployment.apply()
```

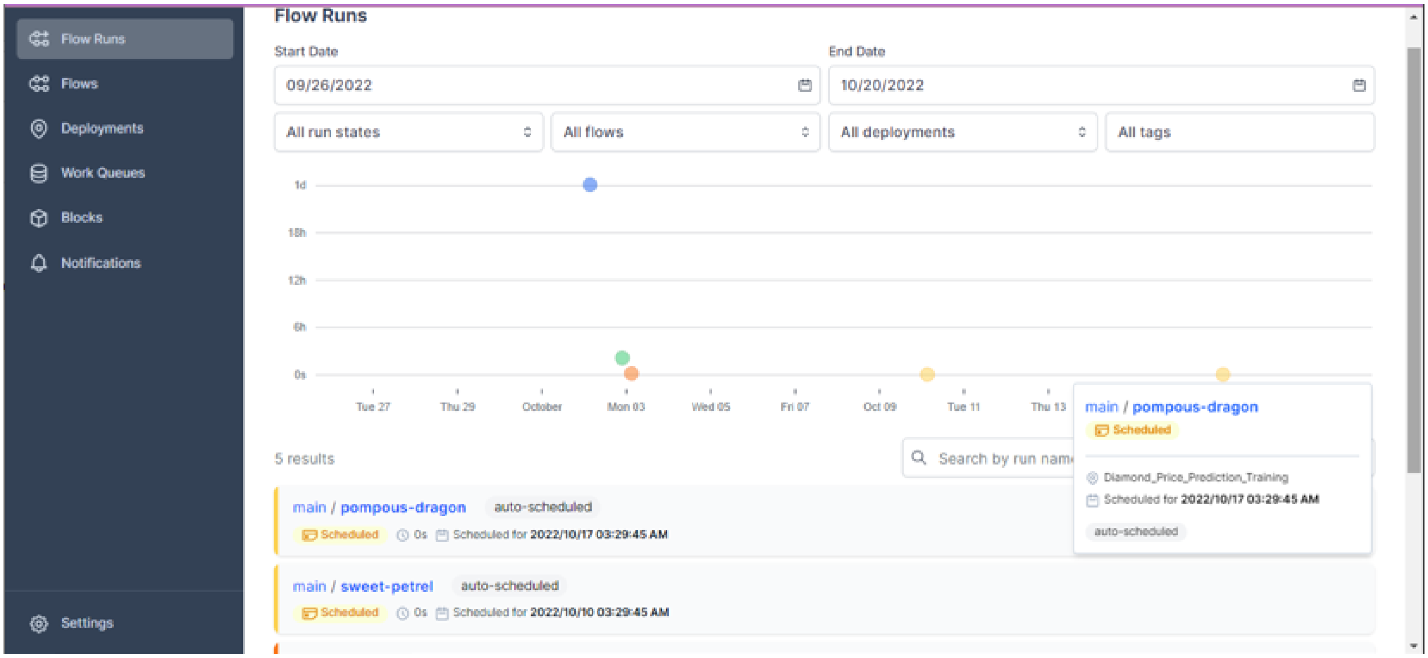
Prefect O/p screens:



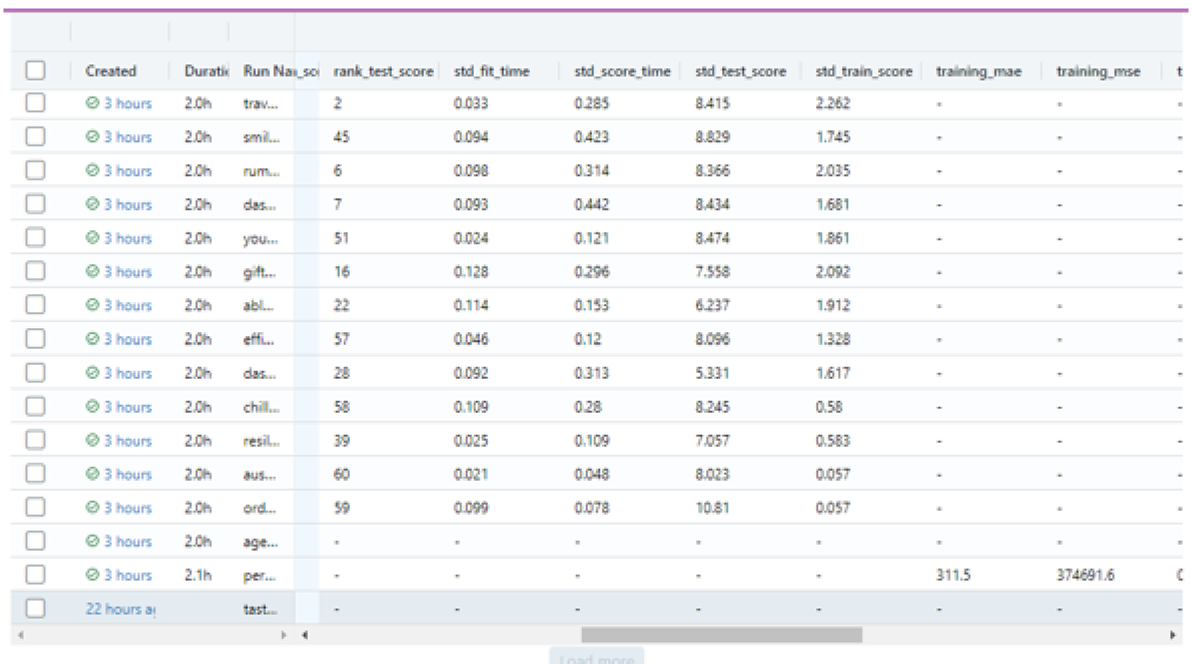



Flow Runs
Flows
Deployments
Work Queues
Blocks
Notifications
Settings
Flow Runs / auburn-magpie
Logs
Task Runs
Sub Flow Runs
Parameters
{
"path": "./data/diamonds.csv"
}
Completed
2h 4m
main
2022/10/02 10:04:53 PM (4s late)
Flow Run ID
ce45494e-5ba0-40d9-8c3d-174286746314
Flow ID
eb4557bc-989d-4081-b320-f9cd2ab270d2
Created
2022/10/02 10:04:49 PM
Updated
2022/10/03 12:09:13 AM
Activate Windows
Go to Settings to activate Windows.


Flow Runs
Flows
Deployments
Work Queues
Blocks
Notifications
Settings
Flow Runs / meticulous-partridge / Radar
Activate Windows
Go to Settings to activate Windows.

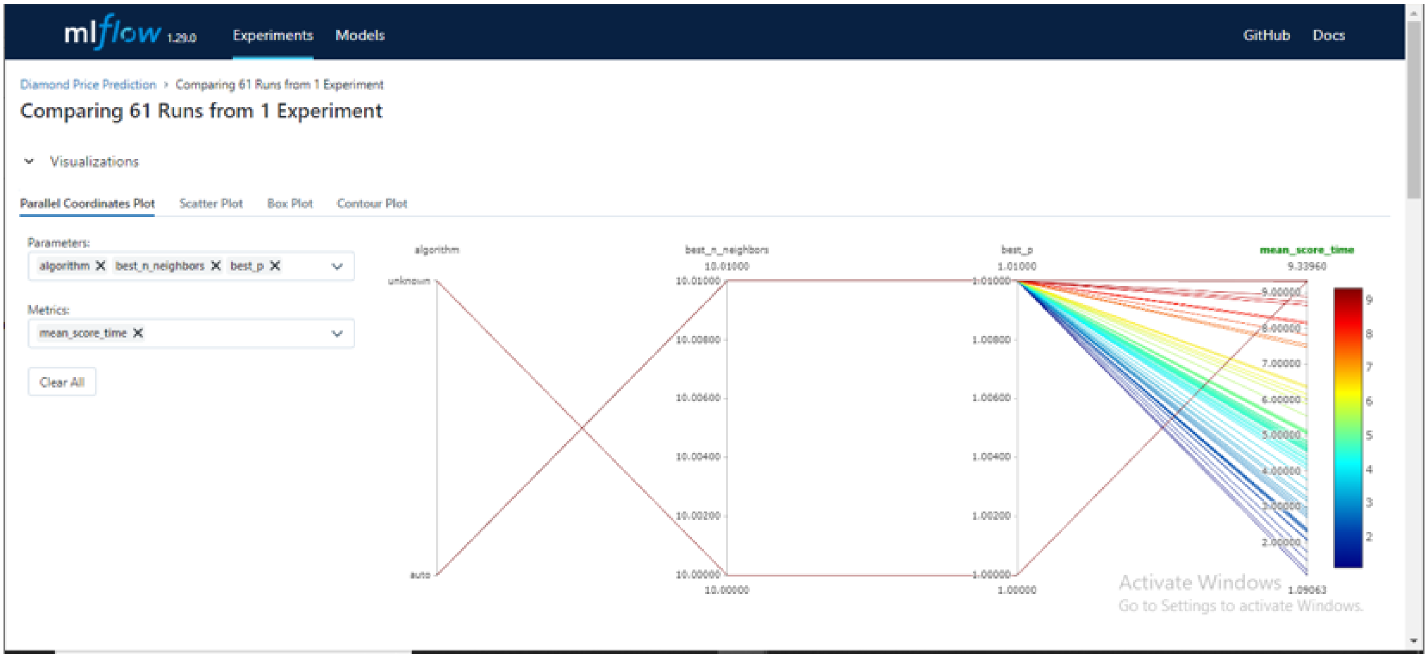


MLFlow O/P Screens:



Created	Duratic	Run Na	sc	rank_test_score	std_fit_time	std_score_time	std_test_score	std_train_score	training_mae	training_mse	t
3 hours	2.0h	trav...		2	0.033	0.285	8.415	2.262	-	-	-
3 hours	2.0h	smil...		45	0.094	0.423	8.829	1.745	-	-	-
3 hours	2.0h	rum...		6	0.098	0.314	8.366	2.035	-	-	-
3 hours	2.0h	das...		7	0.093	0.442	8.434	1.681	-	-	-
3 hours	2.0h	you...		51	0.024	0.121	8.474	1.861	-	-	-
3 hours	2.0h	gift...		16	0.128	0.296	7.558	2.092	-	-	-
3 hours	2.0h	abl...		22	0.114	0.153	6.237	1.912	-	-	-
3 hours	2.0h	effi...		57	0.046	0.12	8.096	1.328	-	-	-
3 hours	2.0h	das...		28	0.092	0.313	5.331	1.617	-	-	-
3 hours	2.0h	chill...		58	0.109	0.28	8.245	0.58	-	-	-
3 hours	2.0h	resil...		39	0.025	0.109	7.057	0.583	-	-	-
3 hours	2.0h	aus...		60	0.021	0.048	8.023	0.057	-	-	-
3 hours	2.0h	ord...		59	0.099	0.078	10.81	0.057	-	-	-
3 hours	2.0h	age...		-	-	-	-	-	-	-	-
3 hours	2.1h	per...		-	-	-	-	-	311.5	374691.6	C
22 hours a		tast...		-	-	-	-	-	-	-	-

Load more


mlflow 1.29.0
Experiments
Models
GitHub
Docs
Diamond Price Prediction › Comparing 61 Runs from 1 Experiment
Comparing 61 Runs from 1 Experiment
Visualizations
Parallel Coordinates Plot
Scatter Plot
Box Plot
Contour Plot
Parameters:
algorithm
best_n_neighbors
best_p
Metrics:
mean_score_time
Clear All
algorithm
best_n_neighbors
best_p
mean_score_time
unknown
auto
Activate Windows
Go to Settings to activate Windows.


Diamond Price Prediction › Comparing 62 Runs from 1 Experiment
Comparing 62 Runs from 1 Experiment
Visualizations
Parallel Coordinates Plot
Scatter Plot
Box Plot
Contour Plot
Parameters:
algorithm
best_n_neighbors
best_p
Metrics:
training_mae
training_rmse
Clear All
algorithm
best_n_neighbors
best_p
training_mae
training_rmse
unknown
auto
Activate Windows
Go to Settings to activate Windows.