

# **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

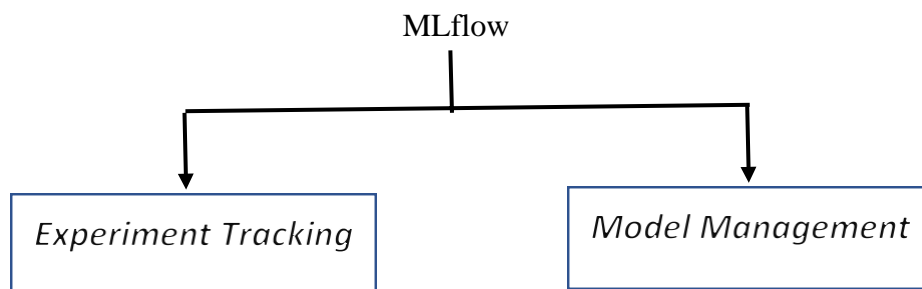
MLflow: Tracking of Experiments, logging or recording all the Experiments.

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.

For the production we choose best one based on the above data.



### **Terminologies:**

1. Experiment -Each trial of some Experiment
2. Run
3. Metadata -All Information related to an Experiment Run (i.e. Tags, Parameters, Data, Train-Test size, Algorithms, Metrics)
4. Artifacts -Output files associated with experiment runs (i.e. Pickle files)

### **Why Track?**

Organization is much easier

Optimization is much more meaningful

**Run below mentioned commands to install mlflow on your system:**

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

[illegible]

```
import mlflow
```

```
mlflow.set_tracking_uri(DATABASE_URI)
mlflow.set_experiment("EXPERIMENT NAME")
```

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

```
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")`

## Running the experiment

```
import mlflow

mlflow.set_tracking_uri("sqlite:///mlflow.db")
mlflow.set_experiment("Diamond Price Prediction")

2022/09/19 13:28:55 INFO mlflow.tracking.fluent: Experiment with name 'Diamond Price Prediction' does not exist
<Experiment: artifact_location='./mlruns/1', experiment_id='1', lifecycle_stage='active', name='Diamond Price F

from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.ensemble import GradientBoostingRegressor

from sklearn import metrics

from pickle import dump
dump(enc, open('pickle_files/Ordinal_Encoder.pkl', 'wb'))
dump(scaler, open('pickle_files/Standard_Scaler.pkl', 'wb'))
```

## Experiment 1 - Training KNN Regressor

```
with mlflow.start_run():
    mlflow.set_tag("dev", "Veena Grace")
    mlflow.set_tag("algo", "KNN")
    # Log the data for each run using log_param, log_metric, log_model
    mlflow.log_param("data-path", "data/diamonds.csv")
    k = 3
    mlflow.log_param("n_neighbors", k)
    knn_regressor = KNeighborsRegressor(n_neighbors=k)
    knn_regressor.fit(X_train_rescaled, y_train)
    y_test_pred = knn_regressor.predict(X_test_rescaled)
    MAE = metrics.mean_absolute_error(y_test, y_test_pred)
    mlflow.log_metric("Mean Absolute error", MAE)
    mlflow.sklearn.log_model(knn_regressor, artifact_path="models")
    mlflow.log_artifact("pickle_files/Standard_Scaler.pkl")
    mlflow.log_artifact("pickle_files/Ordinal_Encoder.pkl")
```

After Executing the above code and open the URL to get the-

## Mlflow Interface For Tracking Experiments

Experiments

☐ Default

☒ Diamond Price Prediction

Diamond Price Prediction

Experiment ID: 1

Description Edit

Refresh

Compare

Delete

Download CSV

Created

All time

Columns

Only show differences

☒ metrics.rmse < 1 and params.model = "tree"

Search

Filter

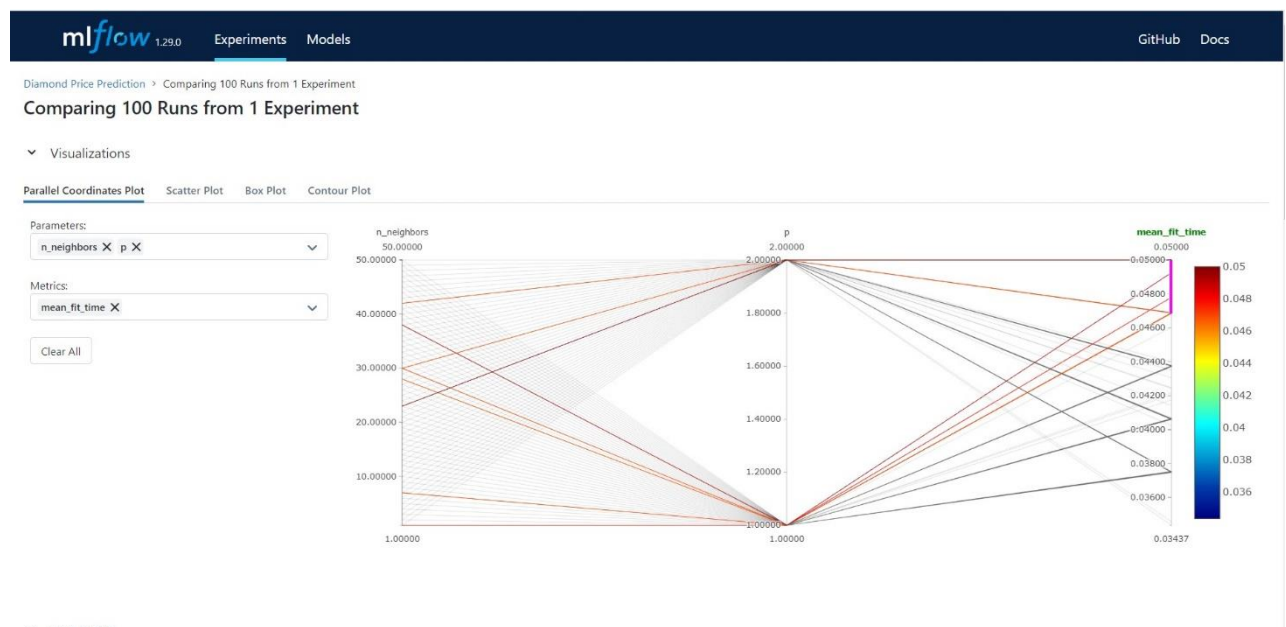
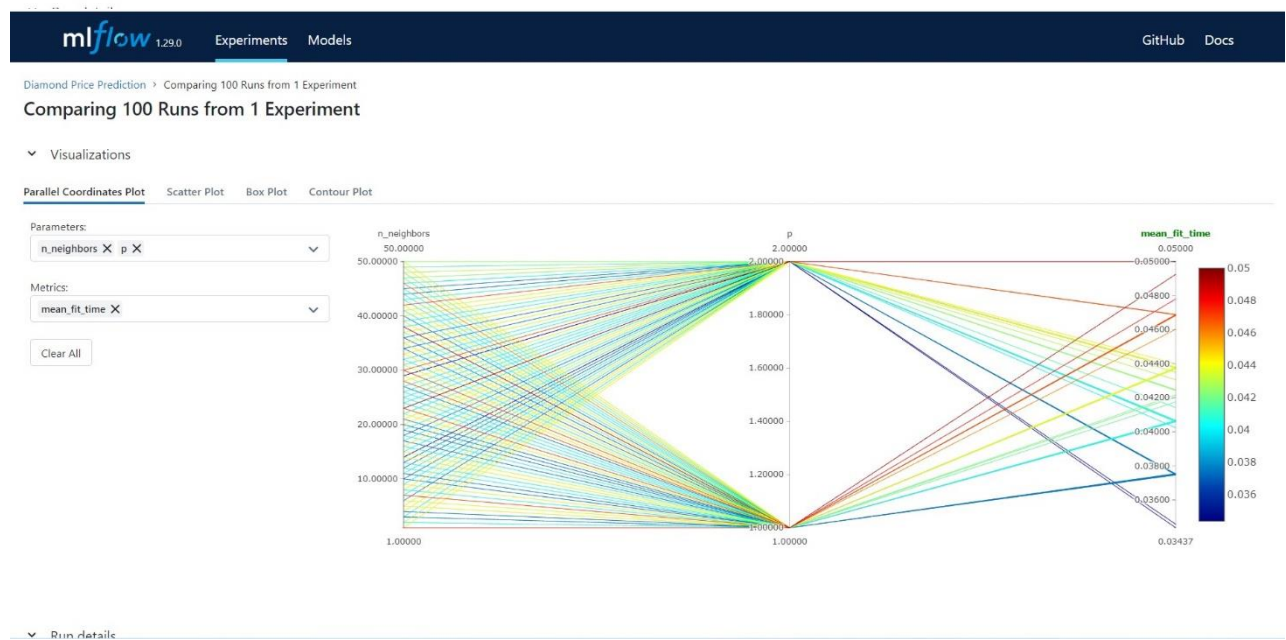
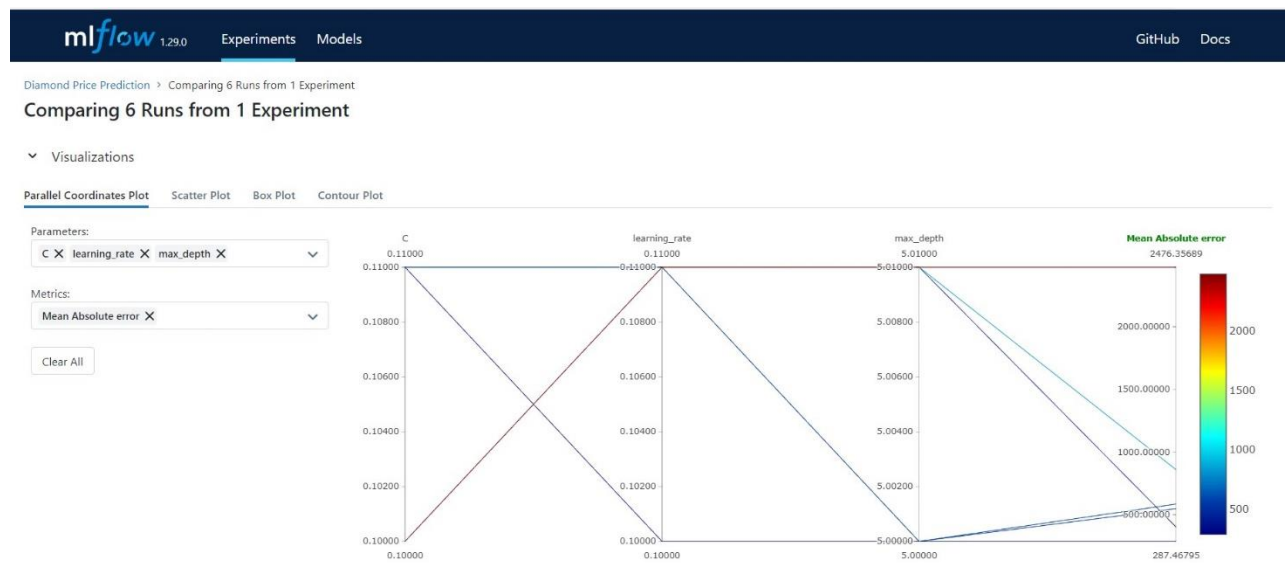
Clear

Showing 6 matching runs

								Metrics	Parameters
	Created	Duration	Run Name	User	Source	Version	Models	Mean Absolute i	C
<input type="checkbox"/>	35 minutes ago	8.0s	lyrical-donk...	DELL	C:\Progra...	-	sklearn	287.5	-
<input type="checkbox"/>	37 minutes ago	2.7min	worried-sho...	DELL	C:\Progra...	-	sklearn	2476.4	0.1
<input type="checkbox"/>	1 hour ago	5.8s	stylish-fowl...	DELL	C:\Progra...	-	sklearn	549.3	-
<input type="checkbox"/>	1 hour ago	3.0s	peaceful-cro...	DELL	C:\Progra...	-	sklearn	585.4	-
<input type="checkbox"/>	1 hour ago	3.4s	secretive-fle...	DELL	C:\Progra...	-	sklearn	859.5	0.1
<input type="checkbox"/>	1 hour ago	10.1s	marvelous-t...	DELL	C:\Progra...	-	sklearn	401.4	-

Load more

## Experiment comparing :



## MLFlow Interface for Model Management

The screenshot shows the MLFlow Model Management interface for a model named 'diamond\_model'. The top navigation bar includes 'mlflow 1.29.0', 'Experiments', and 'Models'. The 'Models' tab is active. Below the navigation bar, the breadcrumb 'Registered Models > diamond\_model' is shown. The model name 'diamond\_model' is displayed, along with its creation and last modification times: 'Created Time: 2022-10-01 19:07:38' and 'Last Modified: 2022-10-01 19:57:14'. There are links for 'Description' and 'Edit'. Below these, there are links for 'Tags'. The 'Versions' section is expanded, showing a table of model versions. The table has columns for 'Version', 'Registered at', 'Created by', 'Stage', and 'Description'. There are three versions listed: 'Version 3' (Archived), 'Version 2' (Staging), and 'Version 1' (Production). Each version has a checkbox and a status icon. At the bottom right, there are navigation buttons: '<', '1', and '>'.

Version	Registered at	Created by	Stage	Description
<input type="checkbox"/> Version 3	2022-10-01 19:08:30		Archived	
<input type="checkbox"/> Version 2	2022-10-01 19:08:15		Staging	
<input type="checkbox"/> Version 1	2022-10-01 19:07:38		Production	

## 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

```
$ 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

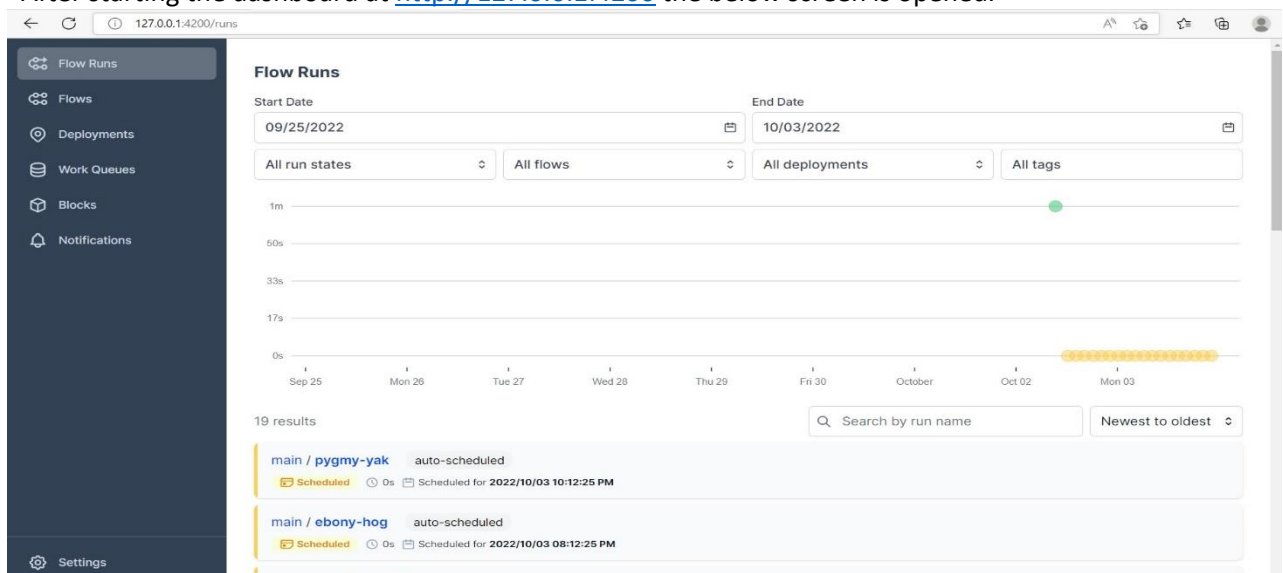
\$ prefect version

## Running Prefect Dashboard

```
$ prefect Orion start
```

[illegible]

After starting the dashboard at <http://127.0.0.1:4200> the below screen is opened.





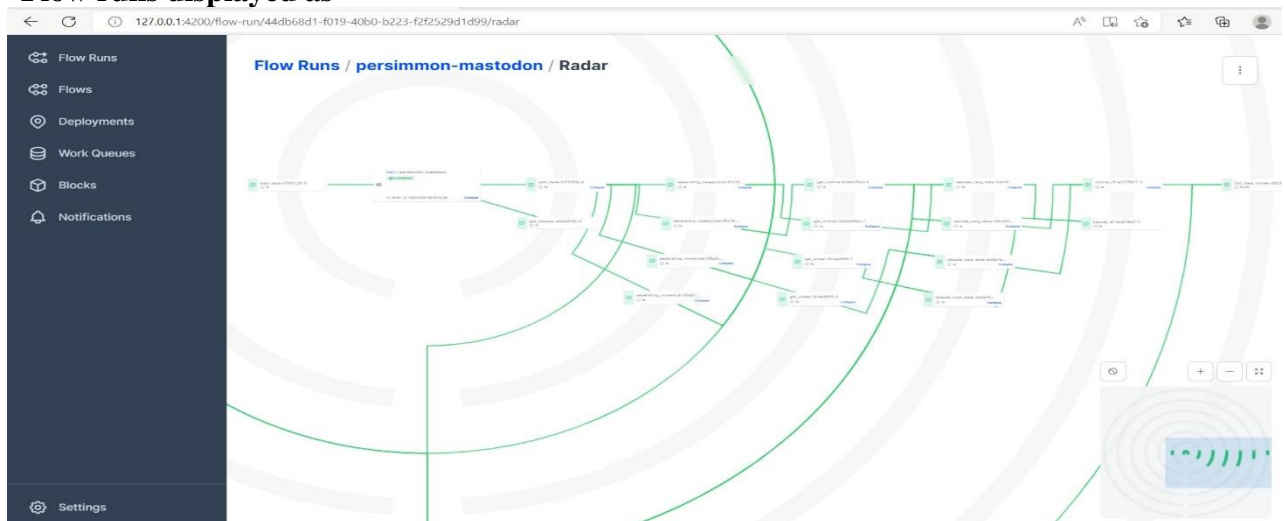
By using this prefect we can also detect the errors in a detailed manner.

The screenshot displays the Prefect UI interface for a specific flow run. The left sidebar contains navigation links for Flow Runs, Flows, Deployments, Work Queues, Blocks, and Notifications. The main content area is titled 'Logs' and shows a detailed log of the flow run's execution. The log entries are as follows:

- Created task run 'load\_data-2ff00c39-0' for task 'load\_data'
- Executing 'load\_data-2ff00c39-0' immediately...
- Created task run 'get\_classes-a0aa979b-0' for task 'get\_classes'
- Executing 'get\_classes-a0aa979b-0' immediately...
- Created task run 'split\_data-b2f518fa-0' for task 'split\_data'
- Executing 'split\_data-b2f518fa-0' immediately...
- Created task run 'seperating\_categorical-81c18760-0' for task 'seperating\_categorical'
- Executing 'seperating\_categorical-81c18760-0' immediately...
- Created task run 'get\_ordinal-9d40d5bd-0' for task 'get\_ordinal'
- Executing 'get\_ordinal-9d40d5bd-0' immediately...
- Created task run 'rescale\_catg\_data-2db16900-0' for task 'rescale\_catg\_data'
- Executing 'rescale\_catg\_data-2db16900-0' immediately...
- Created task run 'seperating\_numerical-08a2c8b5-0' for task 'seperating\_numerical'
- Executing 'seperating\_numerical-08a2c8b5-0' immediately...
- Created task run 'get\_scaler-9c4a96f8-0' for task 'get\_scaler'
- Executing 'get\_scaler-9c4a96f8-0' immediately...

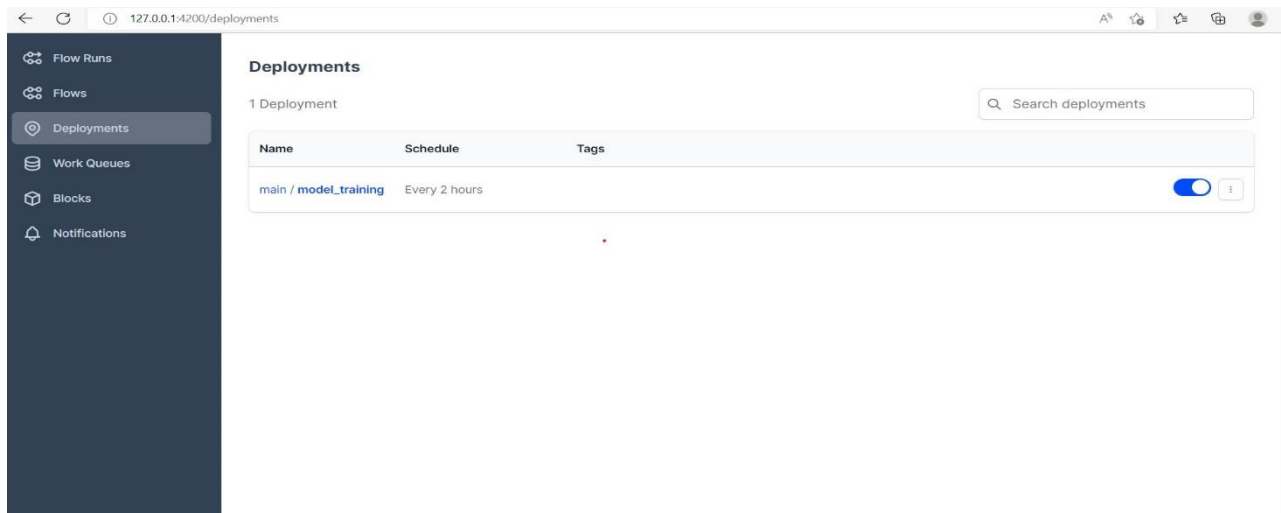
On the right side, there is a summary panel showing the flow run's status as 'Completed' with a duration of '1m 5s'. It also displays the flow ID, flow run ID, and other metadata. A small circular progress indicator is visible in the top right corner.

Flow runs displayed as



## Deployment of Prefect Flow

- `work_queue_name` is used to submit the deployment to the 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.



## Running an Agent

```
$ prefect agent start --work-queue "ml"
```

```
(mlops_env) C:\Users\DELL\ML-project\Orchestrate_ML_Pipeline\version_2>cd ..

(mlops_env) C:\Users\DELL\ML-project\Orchestrate_ML_Pipeline>version_3
'version_3' is not recognized as an internal or external command,
operable program or batch file.

(mlops_env) C:\Users\DELL\ML-project\Orchestrate_ML_Pipeline>cd version_3

(mlops_env) C:\Users\DELL\ML-project\Orchestrate_ML_Pipeline\version_3>python workflow_v3.py

(mlops_env) C:\Users\DELL\ML-project\Orchestrate_ML_Pipeline\version_3>python workflow_v3.py

(mlops_env) C:\Users\DELL\ML-project\Orchestrate_ML_Pipeline\version_3>prefect agent start --work-queue="ml"
Usage: prefect agent start [OPTIONS] [WORK_QUEUE]
Try 'prefect agent start --help' for help.

Error: No such option: --work-queue=ml Did you mean --work-queue?

(mlops_env) C:\Users\DELL\ML-project\Orchestrate_ML_Pipeline\version_3>prefect agent start --work-queue "ml"
Starting v2.4.5 agent with ephemeral API...

PREFECT AGENT

Agent started! Looking for work from queue(s): ml...
```



