



INNOVATION. AUTOMATION. ANALYTICS

PROJECT ON

Experiment Tracking and Model Management
(Sentiment Analysis Project)

Submitted by:

Pandey Abhishek Nath Roy

Intern ID : 1240134

About me

- Hi there! I am **Pandey Abhishek Nath Roy (IN1240134)**, a fresher data enthusiast, currently learning various things to crack an opportunity to go further.
- Apart from this, I possess the problem solving ability and I am good at learning new things that makes me an ideal candidate to follow my dreams.
- I have previously worked as a data science intern at “**Innomatics Research Labs**” and right now, I am doing the internship to refine my skills at their highest level.
- Feel free to reach out! You can do so by following the below links:
[Github @vjabhi000985](#)
[Linkedin](#)



OBJECTIVE OF THE PROJECT

- The objective of this task is to introduce you to MLflow for experiment tracking, model management, and reproducibility in machine learning projects for the “**Sentiment Analysis Project**”.

MLFLOW

- “MLflow” is an open source platform for managing the machine learning lifecycle.
- **What it does :-**
 - ☐ ***Tracks experiments:*** Logs parameters, metrics, and artifacts (models, code) during training runs.
 - ☐ ***Enables reproducibility:*** Compares experiments and facilitates replicating successful runs.
 - ☐ ***Model registry:*** Stores, versions, manages, and deploys trained models.
- **Benefits :-**
 - ☐ Improves Collaboration.
 - ☐ Streamline deployments and Automates workflows.

Integration of MLflow into Projects

```
# Install the required packages
$ pip3 install mlflow dagshub --quiet

import mlflow

mlflow.set_experiment("flipkart_review_sentiment_prediction")

# Start the experiment run
with mlflow.start_run() as run:
    %time grid_search.fit(X_train, y_train)
```

MLflow Dashboard

The screenshot shows the MLflow 2.7.1 Experiments dashboard. The top navigation bar includes the MLflow logo, version 2.7.1, and tabs for Experiments and Models. On the right, there are links to GitHub and Docs. The main section is titled 'flipkart_review_sentiment_prediction' and includes a 'Share' button and a 'Provide Feedback' link. Below the title, the Experiment ID is 1 and the Artifact Location is mlflow-artifacts:/54fdda955c894332ae90daa70f029d72. A 'Description' section is visible with an 'Edit' link. A search bar contains the query 'metrics.rmse < 1 and params.model = "tree"'. Below the search bar, there are filters for 'State: Active', 'Sort: Created', and 'Columns'. A '+ New run' button is on the right. The 'Table' tab is selected, showing a list of runs. The table has columns for Run Name, Created, Dataset, Duration, and Source. Four runs are listed: svc_run, knn_run, decision_tree_run, and naive_bayes_run. The bottom of the table shows '4 matching runs'.

Experiments + □

Search Experiments

☒ flipkart_review_sentiment... ✎ 🗑

flipkart_review_sentiment_prediction 🔗 [Provide Feedback](#) 🔗 [Share](#)

Experiment ID: 1 Artifact Location: mlflow-artifacts:/54fdda955c894332ae90daa70f029d72

> Description [Edit](#)

🔍 metrics.rmse < 1 and params.model = "tree" ⓘ Time created ▾

State: Active ▾ Sort: Created ▾ Columns ▾ ⋮ 🔍 📄 🔄 [+ New run](#)

Table Chart Evaluation **Experimental**

<input type="checkbox"/>	<input type="checkbox"/>	Run Name	Created	Dataset	Duration	Source
<input type="checkbox"/>	<input type="checkbox"/>	svc_run	✓ 22 minutes ago	-	11.2s	colab_
<input type="checkbox"/>	<input type="checkbox"/>	knn_run	✓ 25 minutes ago	-	11.7s	colab_
<input type="checkbox"/>	<input type="checkbox"/>	decision_tree_run	✓ 29 minutes ago	-	11.3s	colab_
<input type="checkbox"/>	<input type="checkbox"/>	naive_bayes_run	✓ 29 minutes ago	-	13.7s	colab_

< > 4 matching runs

Demonstration of Parameters, Metrics and Artifacts

mlflow 2.7.1 Experiments Models GitHub Docs

flipkart_review_sentiment_prediction >

svc_run

Run ID: d4b05761c417421e964bd223762f21b5 Date: 2024-03-28 21:11:55 Source: colab_kernel_launcher.py

User: princeabh00985 Duration: 11.2s Status: FINISHED

Lifecycle Stage: active

> Description Edit

> Datasets

▼ Parameters (3)

Name	Value
algorithm	svc
best_hyperparameter	{'classifier_C': 1, 'classifier_kernel': 'linear'}
hyperparameter_grid	[{'classifier_kernel': 'rbf', 'classifier_C': [0.1, 0.01, 1, 10, 100]}, {'classifier_kernel': 'poly', 'classifier_degree': [2, 3, 4, 5], 'classifier_C': [0.1, 0.01, 1, 10, 100]}, {'classifier_kernel': 'linear', 'classifier_C': [0.1, 0.01, 1, 10, 100]}]

▼ Metrics (5)

Name	Value
------	-------

▼ Artifacts

- svc_model
 - metadata
 - MLmodel
 - conda.yaml
 - python_env.yaml
 - requirements.txt
 - MLmodel
 - conda.yaml
 - model.pkl
 - python_env.yaml
 - requirements.txt

Full Path: mlflow-artifacts/54fdda955c894332ae90daa70f029d72/d4b05761c417421e964bd223762f21b5/artifact...
Registered on 2024/03/28

MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. This model is also registered to the [model registry](#).

Model schema

Input and output schema for your model. [Learn more](#)

Name	Type
No schema. See MLflow docs for how to include input and output schema with your model.	

Make Predictions

Predict on a Spark DataFrame:

```
import mlflow
from pyspark.sql.functions import struct, col
logged_model = 'runs:/d4b05761c417421e964bd223762f21b5/svc_model'

# Load model as a Spark UDF. Override result_type if the model does not return double values.
loaded_model = mlflow.pyfunc.spark_udf(spark, model_uri=logged_model, result_type='double')

# Predict on a Spark DataFrame.
df.withColumn('predictions', loaded_model(struct(*map(col, df.columns))))
```

Predict on a Pandas DataFrame:

```
import mlflow
logged_model = 'runs:/d4b05761c417421e964bd223762f21b5/svc_model'

# Load model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(logged_model)

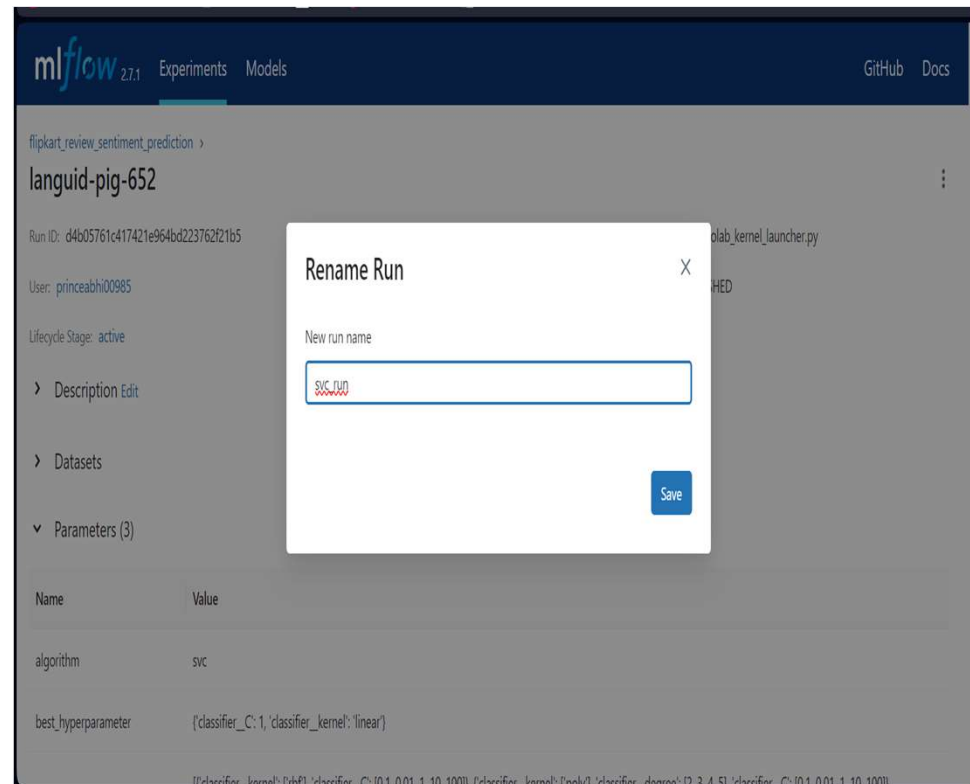
# Predict on a Pandas DataFrame.
import pandas as pd
```

Demonstration of Parameters, Metrics and Artifacts using MLflow tracking API

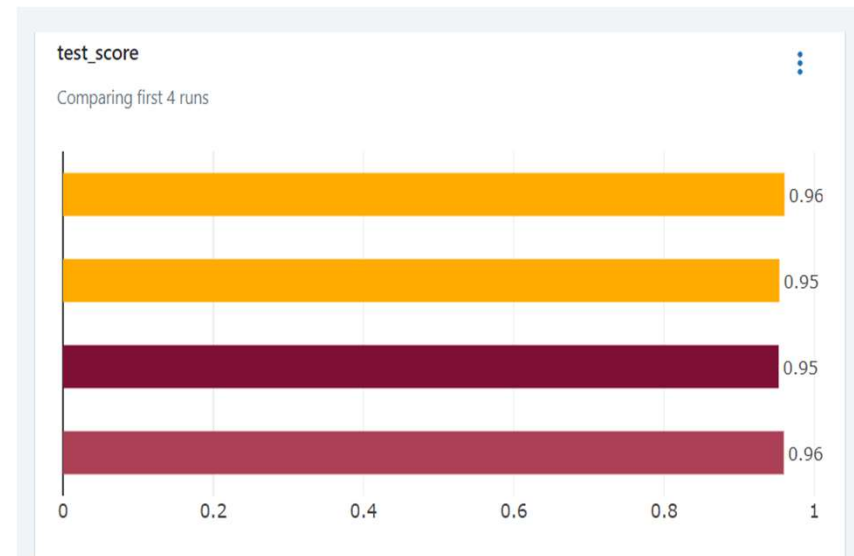
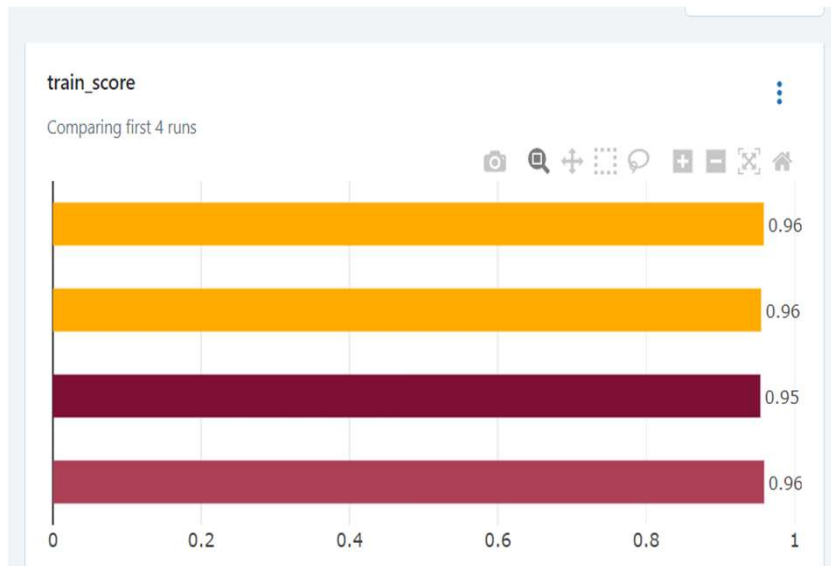
Click on "Experiment Name" -> Run Name -> (scroll down for) -> Parameters, Metrics and Artifacts .

Customizing Mlflow with Run Names

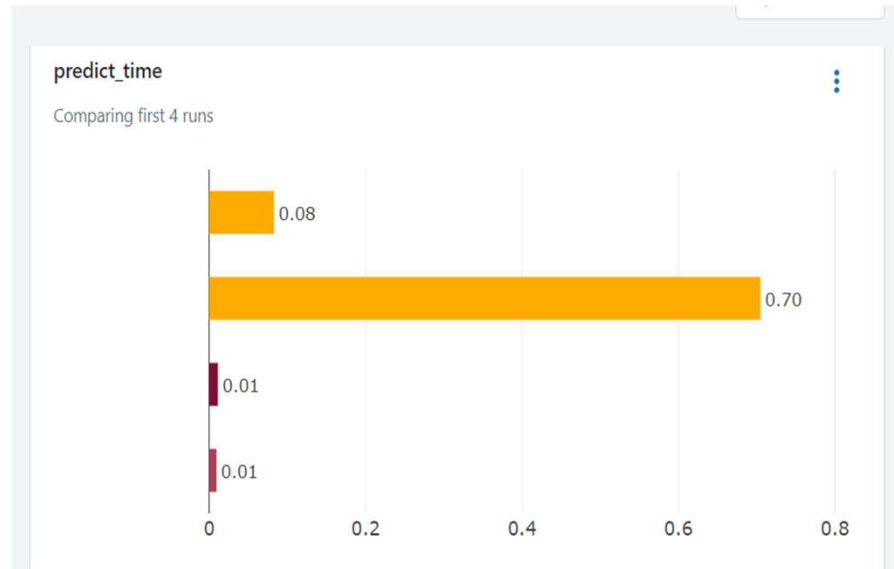
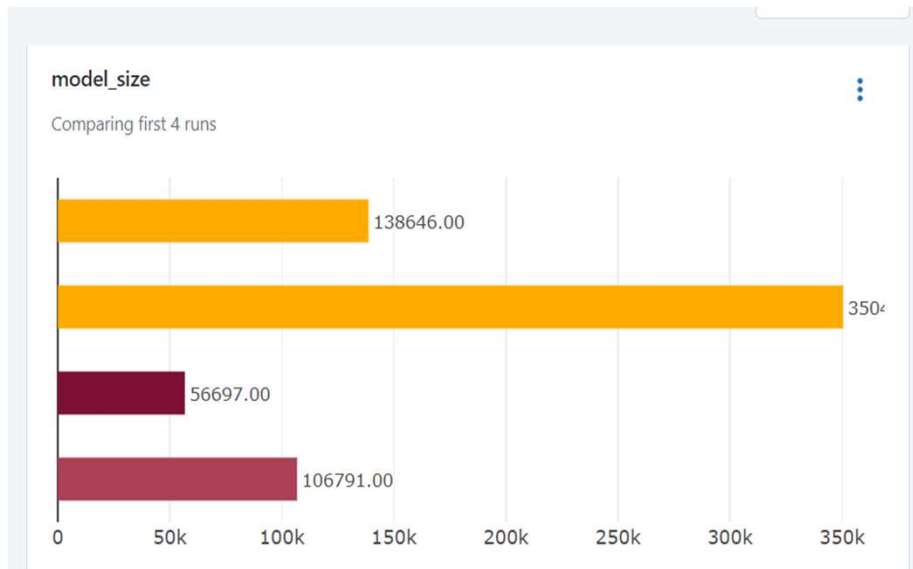
**Click on the run name -> Right corner : 3 dots ->
Rename -> Give the name of run -> Save**



Metric Plots :



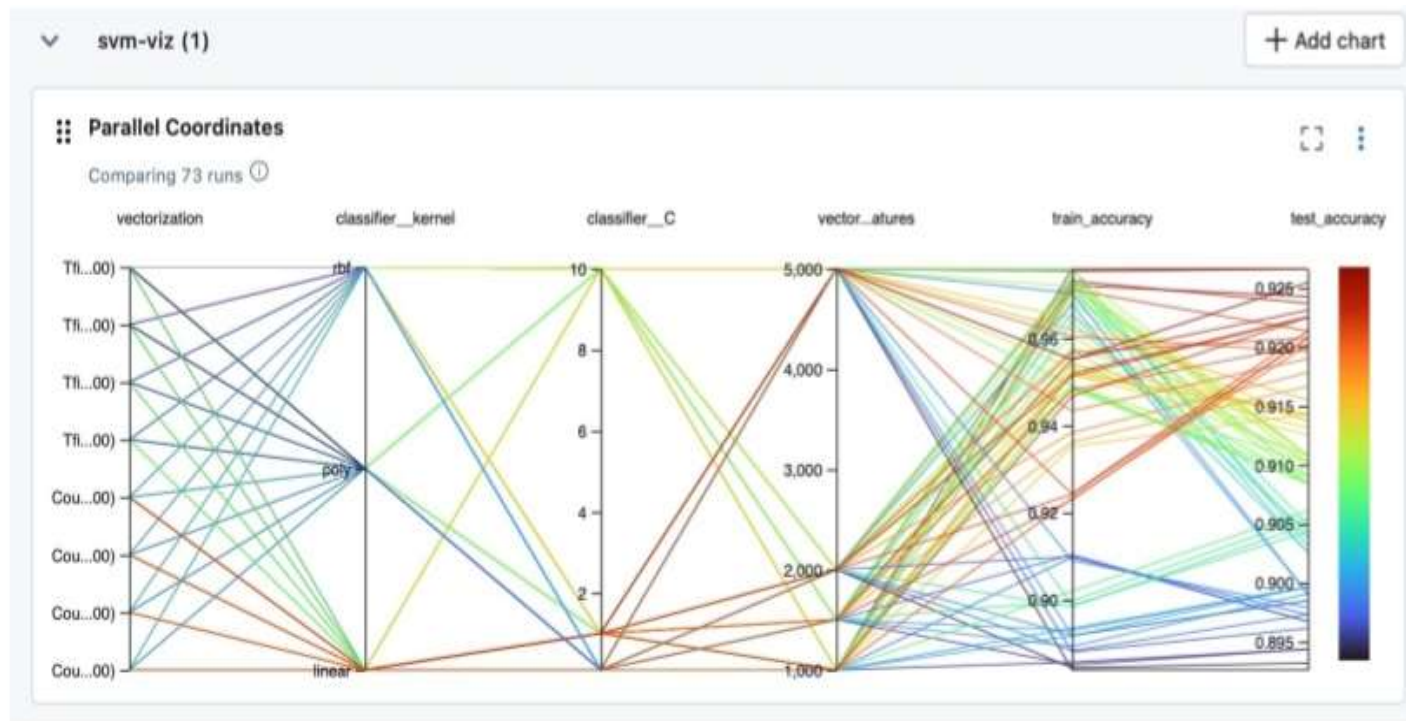
Metric Plots :



Hyperparameter Plots Creation

*Click on charts -> Add Section -> Name hyperparameter with model ->
Add Parallel coordinates chart -> give parameters and metrics of particular run ->
Click on the run-name's "+": This will pop interactive chart.*

Hyperparameter Plots



Hyperparameter Plots



Registering the models

mlflow2.7.1

ExperimentsModels

GitHubDocs

Registered Models >

flipkart_sentiment_analysis

Created Time: 2024-03-28 21:29:10Last Modified: 2024-03-28 21:33:32

> DescriptionEdit

> Tags

> Versions

AllActive 2Compare

<input type="checkbox"/>	Version	Registered at	Created by	Stage	Description
<input type="checkbox"/>	<input checked="" type="checkbox"/> Version 4	2024-03-28 21:31:43		None	
<input type="checkbox"/>	<input checked="" type="checkbox"/> Version 3	2024-03-28 21:31:23		Staging	
<input type="checkbox"/>	<input checked="" type="checkbox"/> Version 2	2024-03-28 21:31:03		Production	
<input type="checkbox"/>	<input checked="" type="checkbox"/> Version 1	2024-03-28 21:29:10		Archived	

<1>

Prefect

Prefect is a Python library that helps you build organized and efficient data pipelines, especially for machine learning. It allows you to:

- **Break down complex tasks:** Prefect lets you divide your data processing or machine learning workflow into smaller, reusable steps called tasks.
- **Manage task order:** It cleverly manages the order in which these tasks run, ensuring everything happens at the right time.
- **Keep an eye on progress:** Prefect monitors your workflows as they run, catching any errors and offering visualizations to help you troubleshoot.

Prefect : Benefits

- **Less code, more reuse:** By using modular tasks, you can write code once and use it in multiple workflows, saving time and effort.
- **Easier maintenance:** Clear organization with tasks makes your workflows easier to understand, modify, and debug later on.
- **Scales as you grow:** Prefect can handle large and complex workflows by allowing tasks to be run on multiple machines.

Prefect : Installation and First run

- Run the following commands to install the prefect and to run it.

☐ `$ pip install prefect`

☐ `$ prefect server start`

```
PS C:\Users\vjavi\Desktop\my_flask_projects\MLflow_Prefect_Orchestration> prefect server start
```

```

  _____
 /         \
|  _   _   |
| /_  _/_  |
|_|_|_|_|_|_|

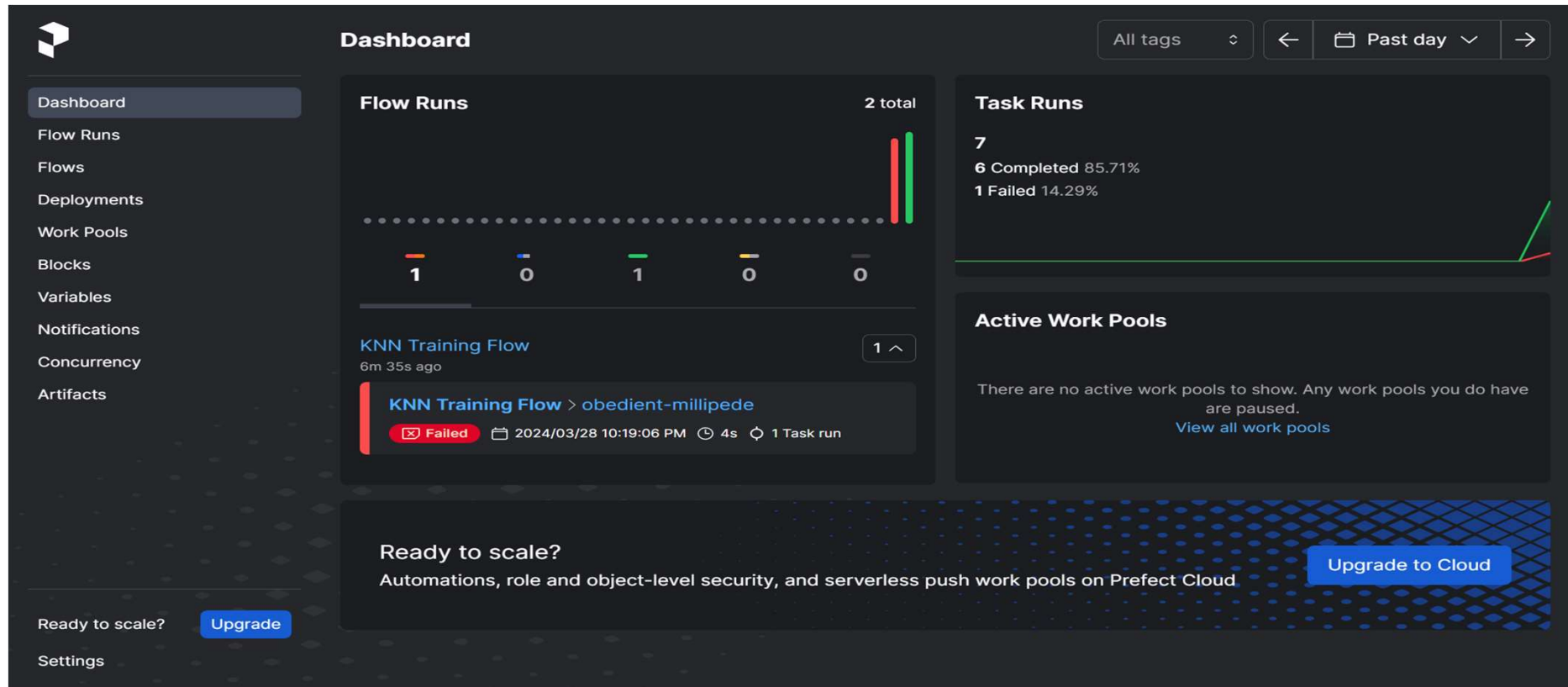
```

Configure Prefect to communicate with the server with:

```
prefect config set PREFECT_API_URL=http://127.0.0.1:4200/api
```

This type of prompt
will appear.

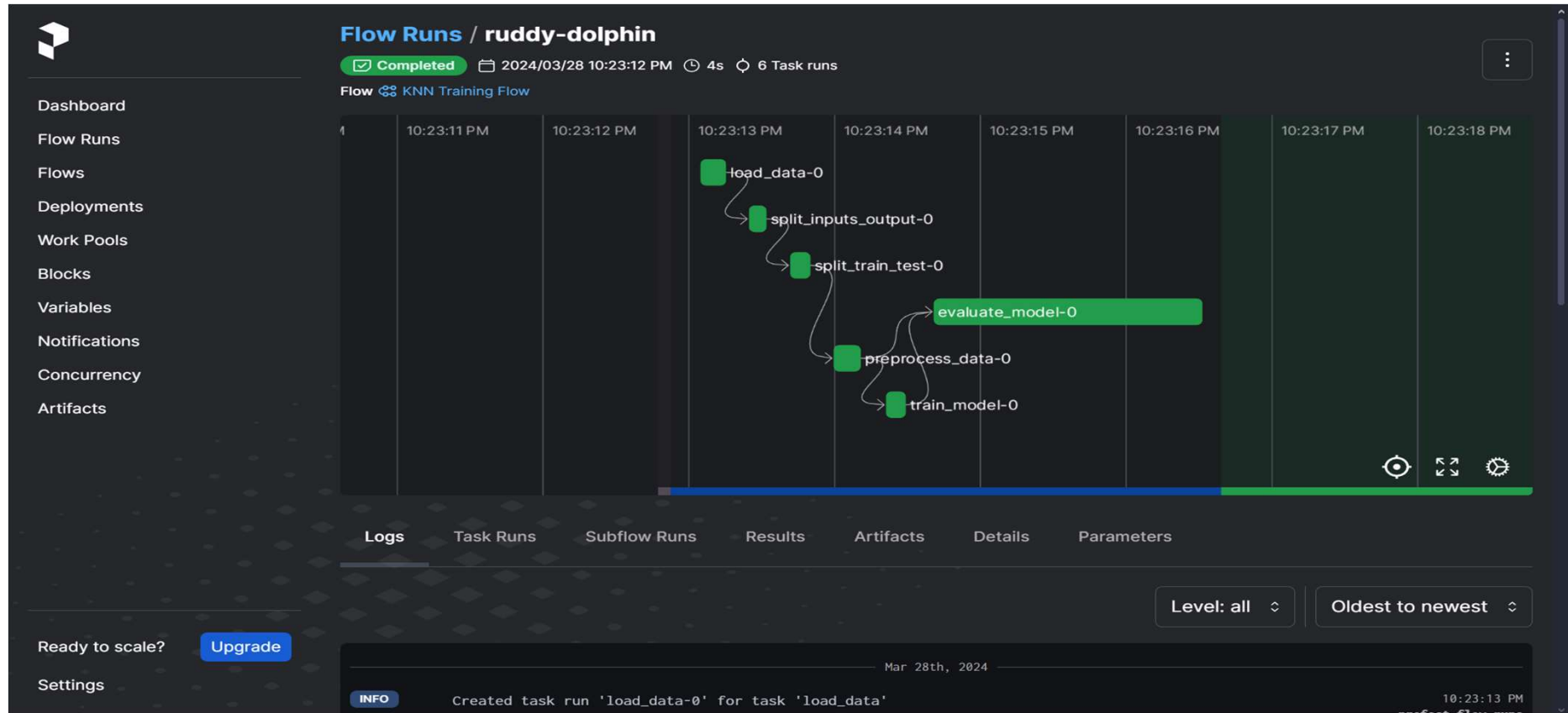
Prefect Dashboard



Building a Prefect workflow

- ✓ **Import Prefect Models *(step 1)*.**
- ✓ **Define Prefect Tasks *(step 2)*.**
- ✓ **Define Prefect Flow *(step 3)*.**
- ✓ **Run Prefect Workflow *(step 4)*.**

Prefect workflow



THANK
YOU

