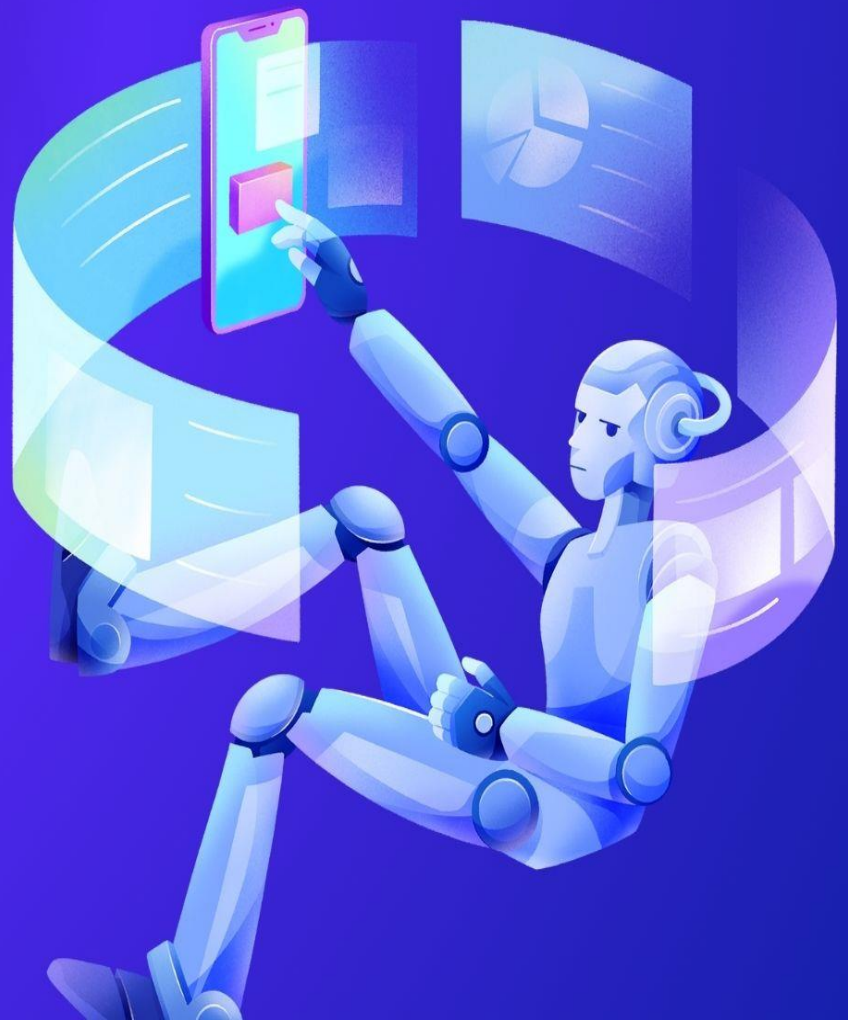
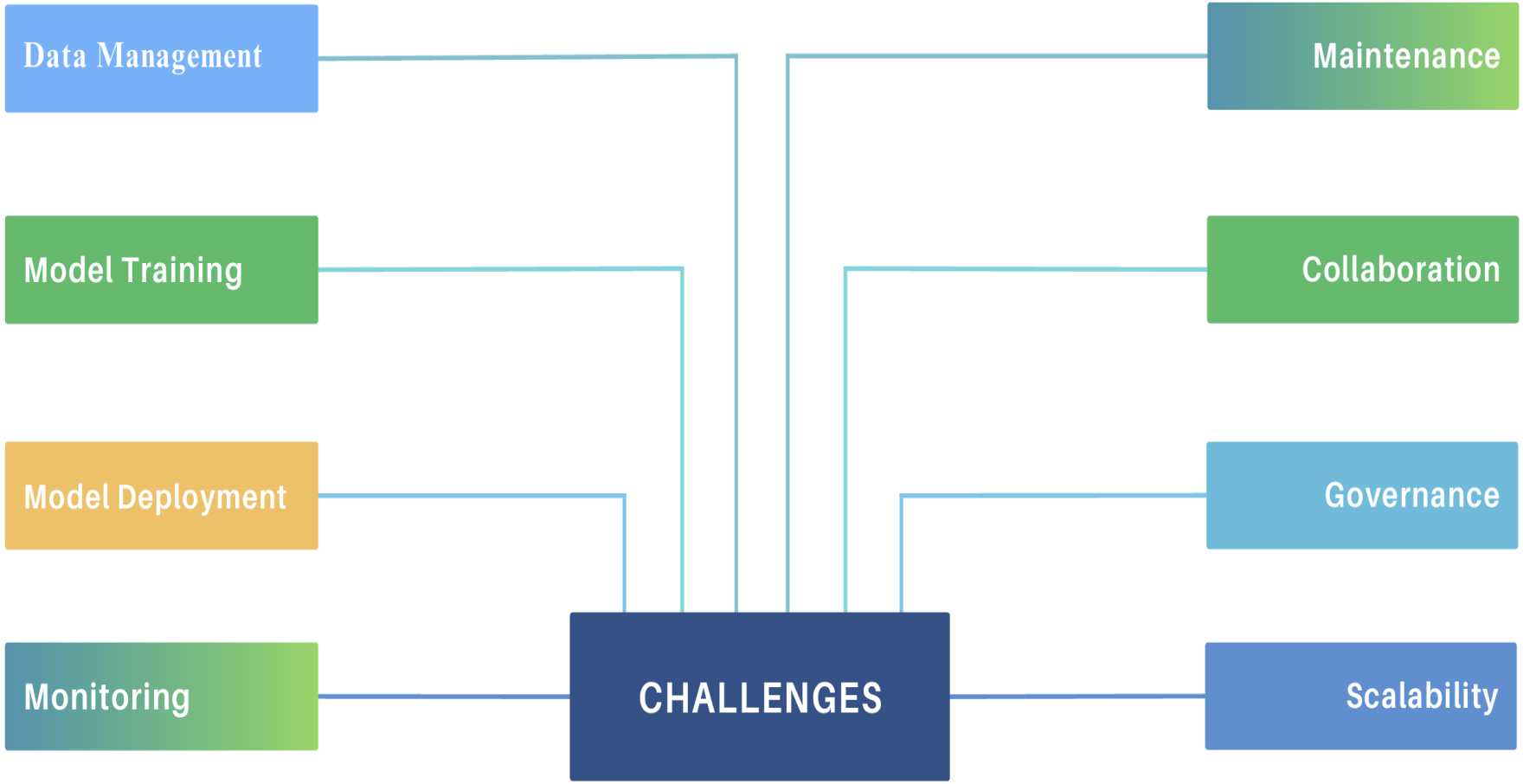
MLOPS



Challenges in ML Lifecycle



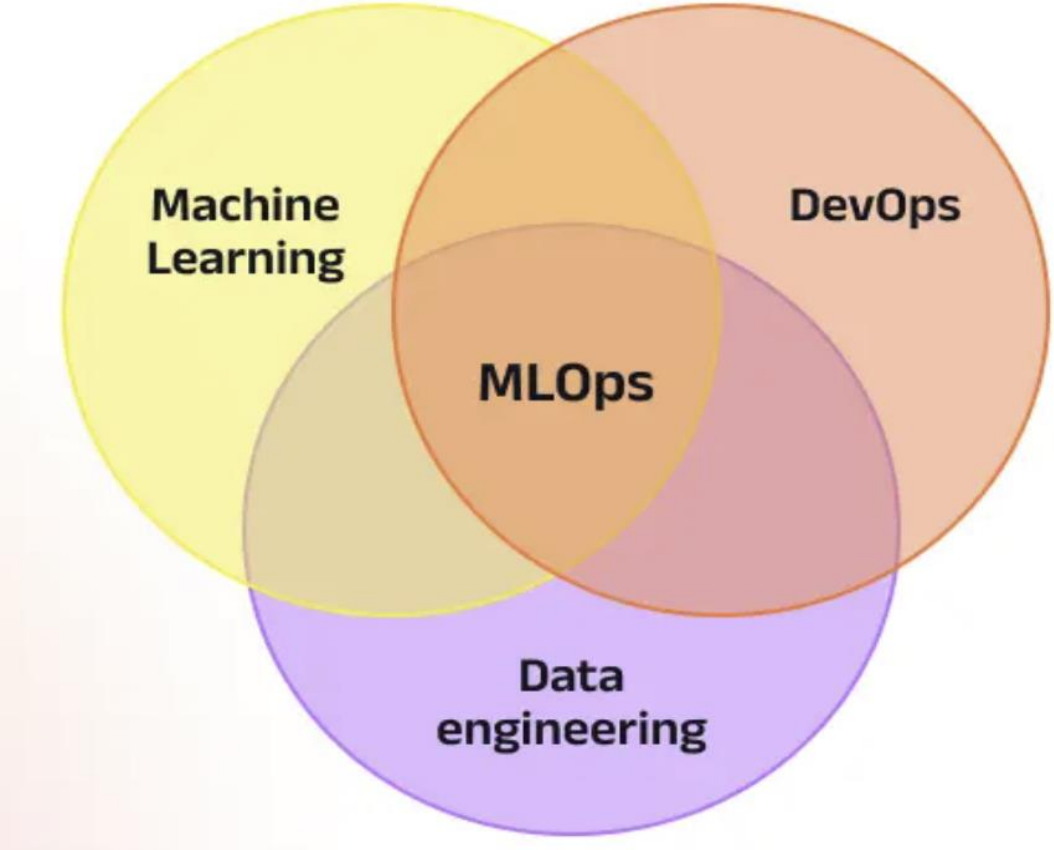
Tools and Techniques for MLOps

Tool / Platform	Purpose
Git	Version control for code.
DVC (Data Version Control)	Version control for datasets.
MLflow	**Model versioning and managing the ML lifecycle.**
Docker	Containerization of ML models and environments.
Kubernetes	Orchestration of containerized applications.
Jenkins / GitLab CI	Automating the testing and deployment of ML models.
Prometheus / Grafana	Monitoring model performance and tracking data drift.
Apache Airflow	Orchestrating and scheduling complex ML workflows.
Kubeflow	Managing and deploying ML workflows on Kubernetes.
Weights & Biases	Experiment tracking and management.
AWS / Google Cloud / Azure	Cloud-based services for ML operations and hosting.

Agenda

- MLOps Vs DevOps
- ML Engineer Hierarchy
- Mlflow
- Components of Mlflow
- Environment Setup
- Lab 1-Using MLflow to track CPU utilization and memory usage(System Monitoring)
- Lab 2:Track scikit-learn model training with MLflow
- Lab 3:Track Keras model training with MLflow

MlOps Vs DevOps



DevOps

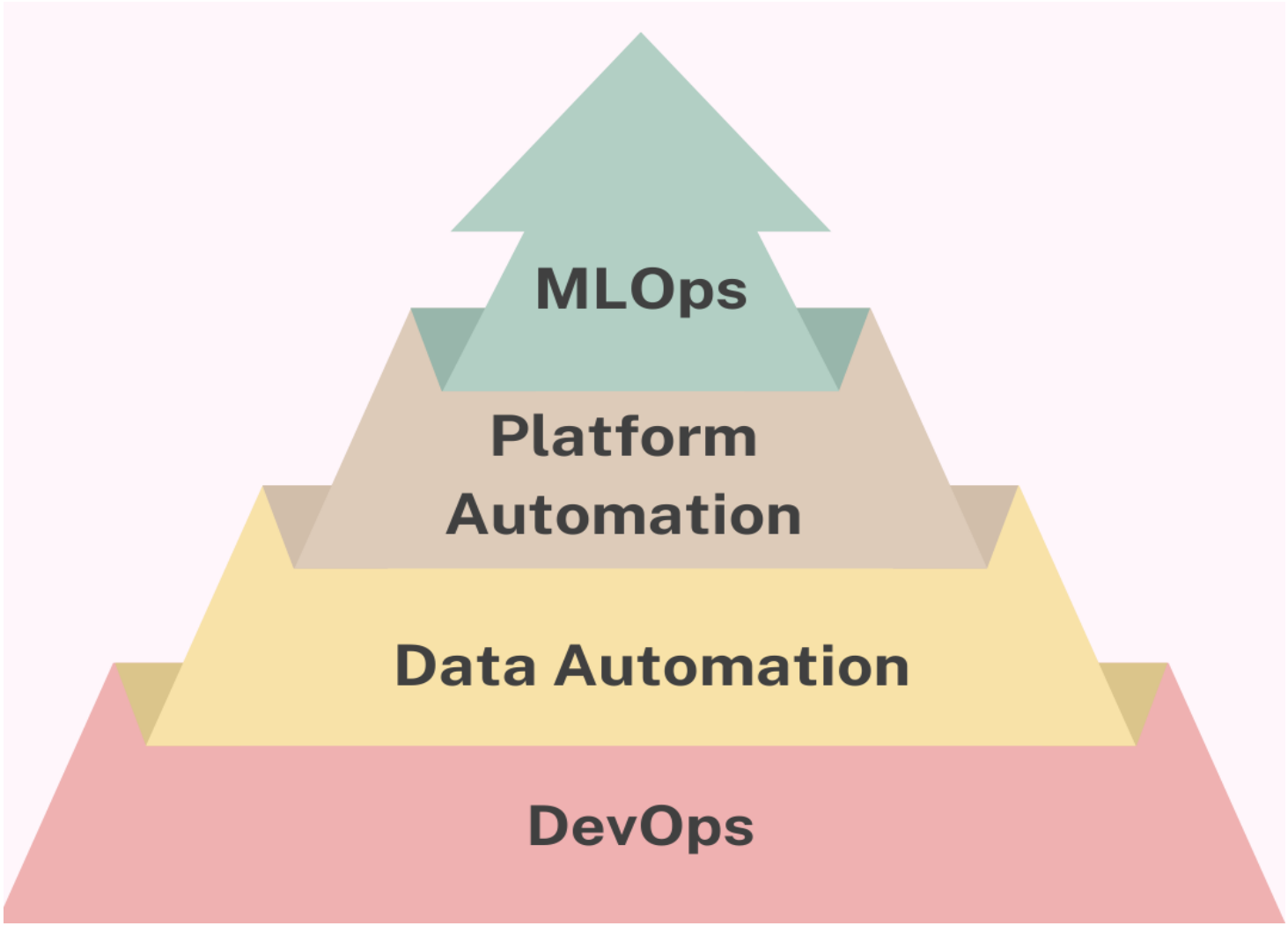
- Focuses on the Software development lifecycle (SDLC).
- Involves continuous integration and continuous deployment (CI/CD) of software applications

Jenkins

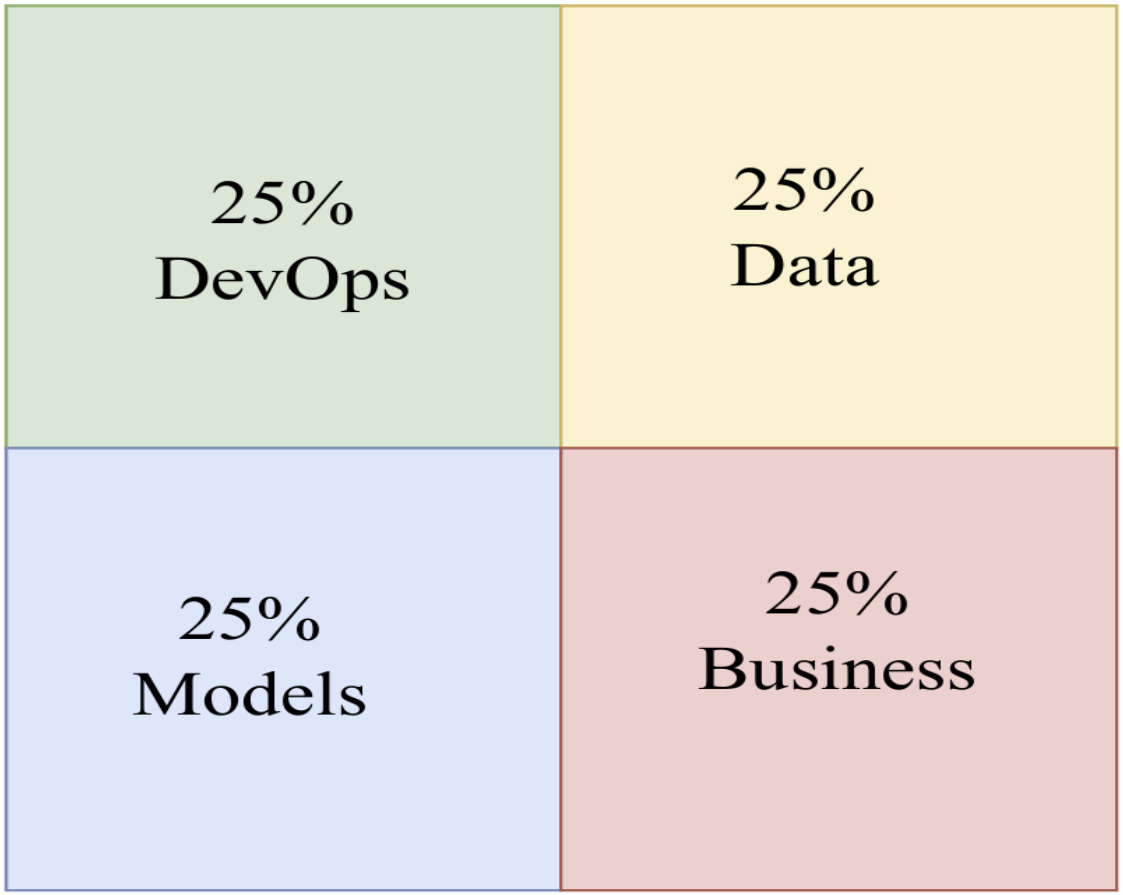
MLOps

- Focuses on the Machine learning lifecycle, including data collection, model training, validation, deployment, and monitoring

ML Engineer Hierarchy



Rule of 25%



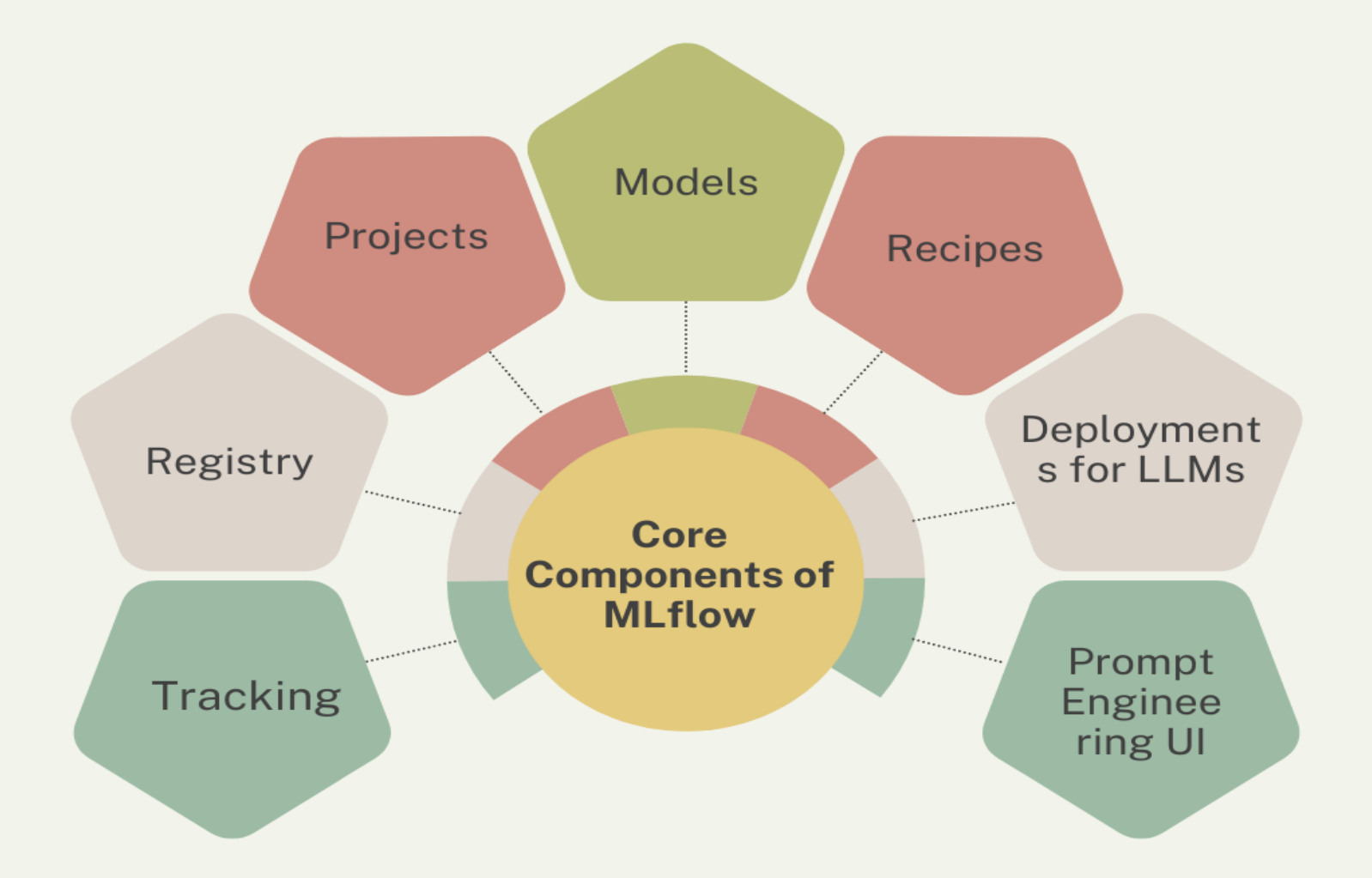

mlflow™

Mlflow

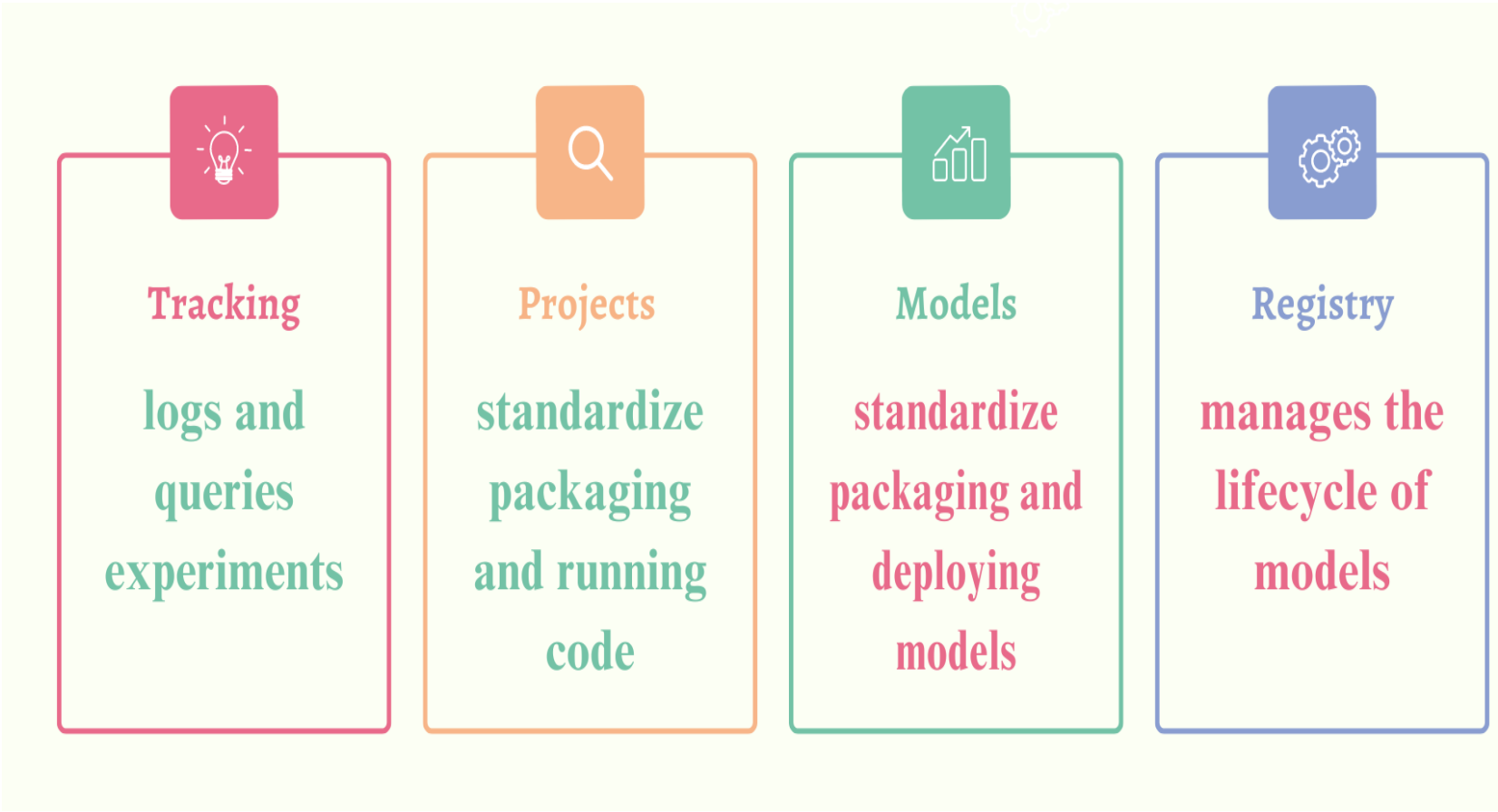
MLflow is an open-source platform designed to manage machine learning workflows, widely utilized by MLOps teams and data scientists.

It provides tools for experiment tracking, model management, and deployment

Core Components of MLflow



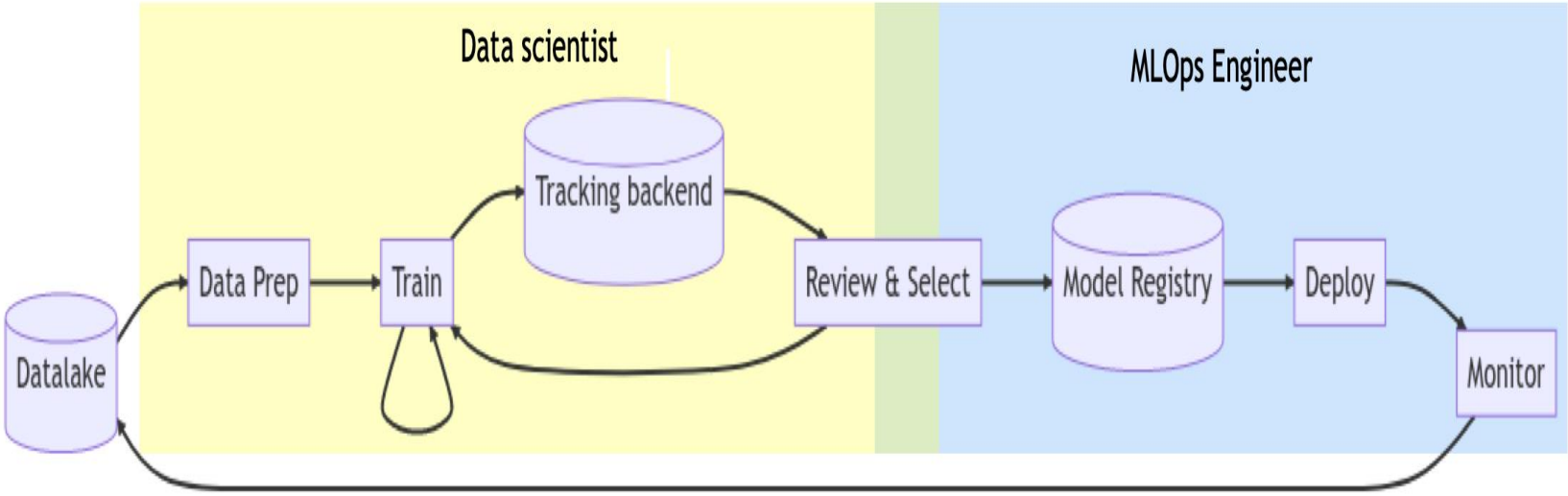
Key Components of MLflow



MLflow Tracking

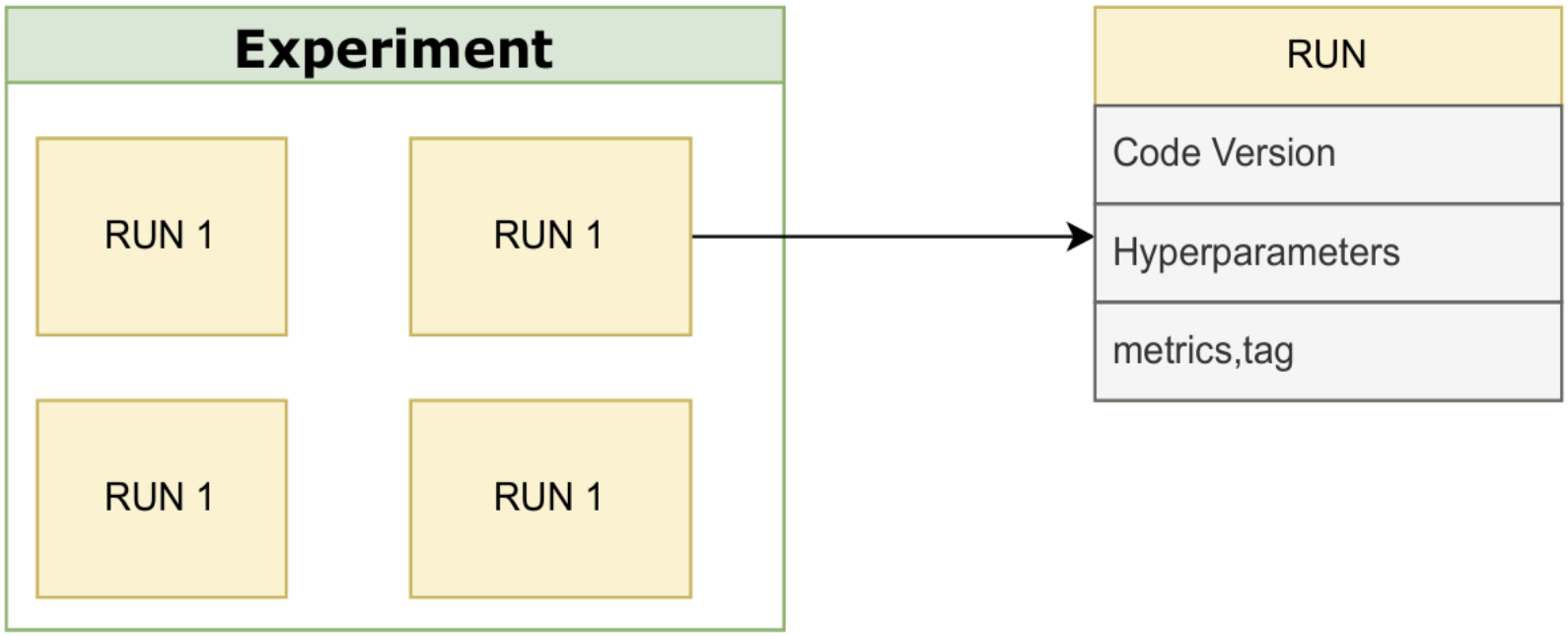
- MLflow Tracking allows data scientists to work with experiments.
- MLflow Tracking is built around runs, that is, executions of code for a data science task.

MLflow Tracking



Experiment Tracking Capabilities

- Logging and Querying Experiments
- UI for Visualization
- Centralized Server
- APIs and Libraries Support

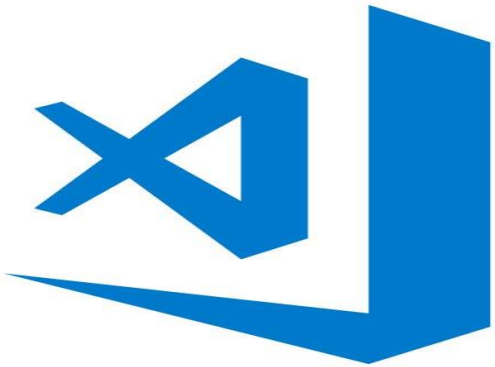

Experiment
RUN 1
RUN 1
RUN 1
RUN 1
RUN
Code Version
Hyperparameters
metrics,tag

Environment Setup

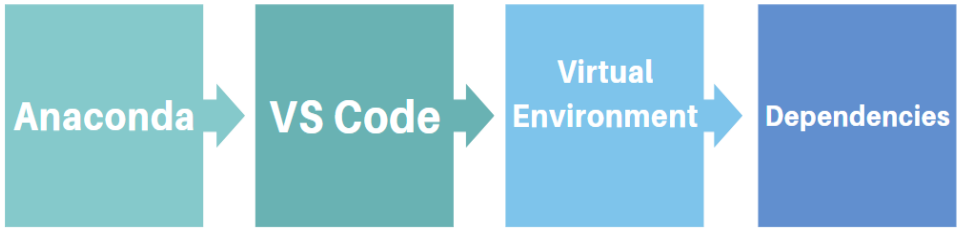
Download & Install

Anaconda

Installing Anaconda on Windows (youtube.com)



Visual Studio Code



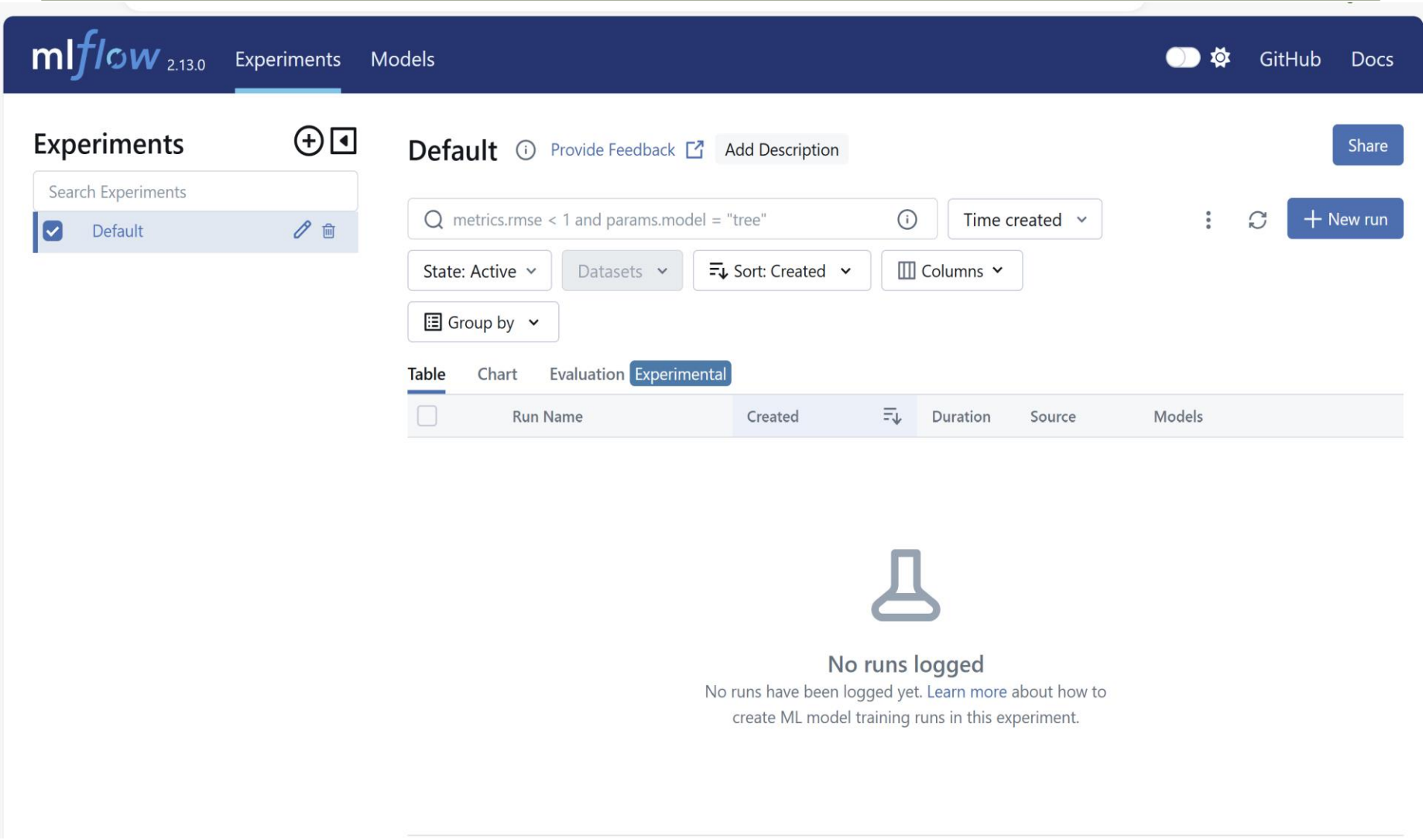
Virtual Environment

conda create --name mlops python=3.11

Dependencies

pip install -r requirements.txt

mlflow ui


mlflow 2.13.0
Experiments
Models
GitHub
Docs
Experiments
Search Experiments
Default
Default
Provide Feedback
Add Description
Share
metrics.rmse < 1 and params.model = "tree"
Time created
New run
State: Active
Datasets
Sort: Created
Columns
Group by
Table
Chart
Evaluation
Experimental
Run Name
Created
Duration
Source
Models
No runs logged
No runs have been logged yet. Learn more about how to create ML model training runs in this experiment.

Logging functions

Function Name	Purpose
mlflow.set_experiment	Sets the name of the experiment under which to log runs (optional).
mlflow.start_run	Starts a new MLflow run.
mlflow.log_param	Logs a parameter used by the model (e.g., hyperparameters).
mlflow.log_metric	Logs a metric (e.g., RMSE, accuracy) to evaluate the model's performance.
mlflow.log_artifact	Logs an artifact (e.g., a file, plot, or other outputs).
mlflow.sklearn.log_model	Logs a scikit-learn model.

System Monitoring