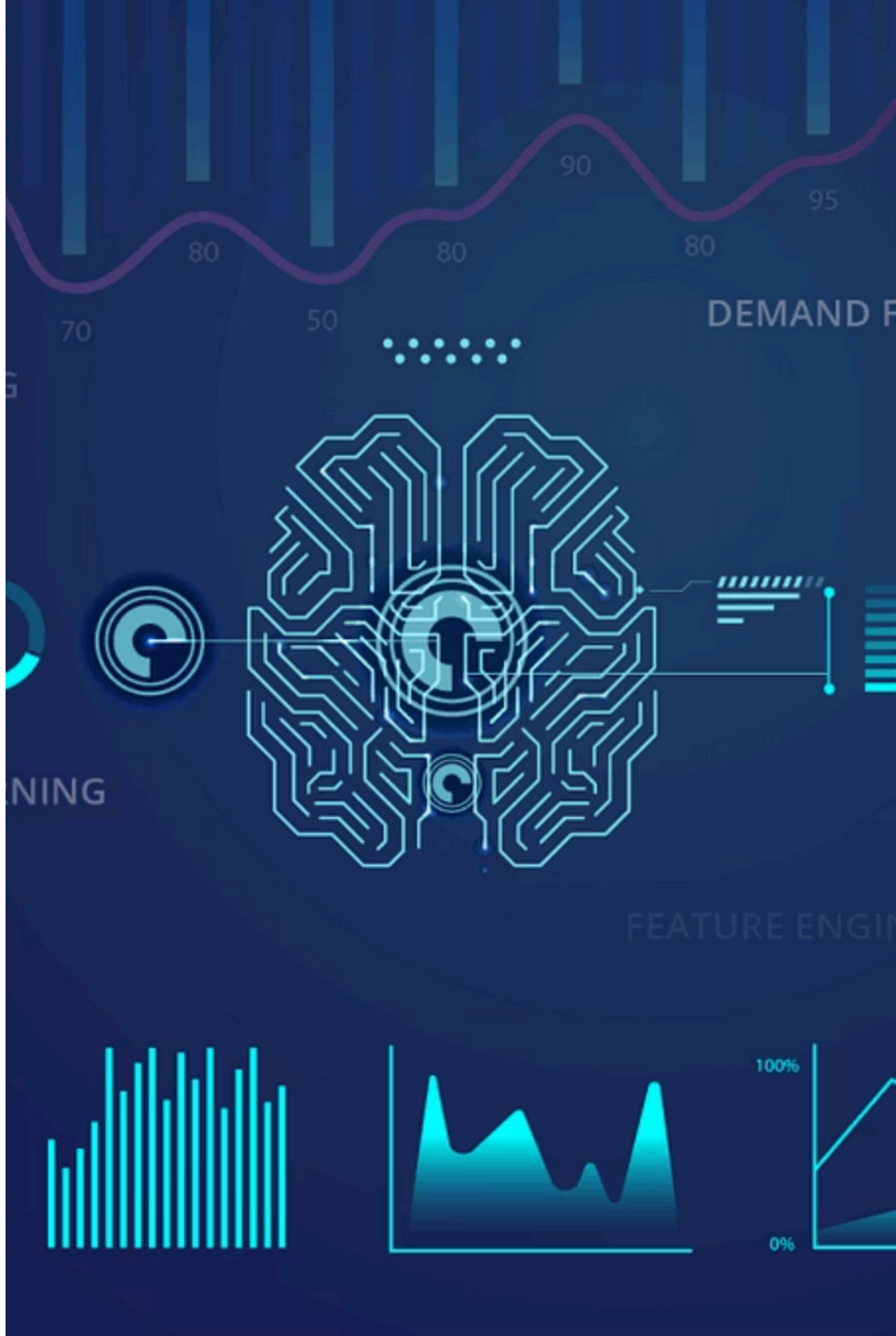
A Specific Machine Learning tool

Mlflow

Presented By : Ahmed Layouni

Overview

- Introduction to MLflow
- MLflow Architecture and Components
- MLflow Vs Other Tools
- Benefits
- Real-World Applications
- Exemple Workflow in MLflow
- Pros/Cons
- Limitations



What is MLflow?

Definition:

MLflow is an open-source platform developed by Databricks to manage the end-to-end machine learning lifecycle, including:

- Experiment tracking
- Model packaging
- Model deployment
- Model registry

Importance: Make ML reproducible, collaborative, and production-ready.

A little comparisation: With and Without MLflow

Without MLflow:

- Wait, which model did we deploy again?”
- “I changed the learning rate, but forgot what it was in the previous run.”
- “We can’t reproduce the result from last month anymore.”
- “Who approved this model to go to production?”

With MLflow:

- Clear logs, version history, and repeatable pipelines
- Teams collaborate without chaos
- Auditable and governed ML workflows

MLflow Architecture and Components

- **MLflow Tracking:**Logs metrics, parameters, artifacts, and source code .

  Key Features:UI for visualization, REST API, SDK integrations

- **MLflow Projects:**Packages ML code in a reproducible format

  Key Features: Standardized format, CLI execution

- **MLflow Models:**Standardizes model packaging and deployment

  Key Features:“Flavors” for deployment tools (SageMaker, Azure)

- **MLflow Registry :**Central hub for managing model versions, stages, and approval workflows

  Key Features: Access control, annotations, webhooks

MLflow Tracking

- **Allows logging and querying experiments**
- **Supports Python, R, Java, and REST API**
- **Stores:**
  - Parameters
  - Metrics
  - Artifacts (e.g., models, images)
  - Source code snapshot
- **Integrates with Jupyter Notebooks, Databricks, and more**

```python
import mlflow

with mlflow.start_run():
    mlflow.log_param("learning_rate", 0.01)
    mlflow.log_metric("accuracy", 0.92)
    mlflow.log_artifact("model.pkl")
```

MLflow Models

- **Supports multiple flavors: scikit-learn, pytorch, keras, xgboost, etc.**
- **Easily deployable:**
  - REST API server
  - AWS SageMaker
  - Azure ML
  - Databricks
- **Automatically includes environment with model**

MLflow Model Registry

- **Manages models lifecycle: Staging → Production → Archived**
- **Supports:**
  - Model versioning
  - Annotations/comments
  - Access control
  - CI/CD integration
- **Helps MLOps teams monitor & govern models.**

MLflow Vs Other Tools

Tool	Primary Focus	Pros	Cons	Best For
MLflow	**End-to-end lifecycle**	- Lightweight, easy setup	- Limited native orchestration	Small to mid-sized teams, hybrid/on-prem deployments
		- Open-source & free	- Basic UI vs. commercial tools	
		- Strong local/cloud flexibility		
Kubeflow	**Kubernetes-native pipelines**	- Scalable on K8s	- Steep learning curve	Large enterprises with K8s expertise
		- Integrated KF components (Katib, KServe)	- Complex K8s dependency	
		- Multi-step pipelines		

MLflow Vs Other Tools

Tool	Primary Focus	Pros	Cons	Best For
Weights & Biases (W&B)	**Experiment tracking & collaboration**	- Superior visualization	- Cloud-centric (SaaS)	Research teams, deep learning focus
		- Artifact lineage	- Pricing scales with usage	
		- Team dashboards		
TensorFlow Extended (TFX)	**Production TF pipelines**	- Tight TensorFlow integration	- TF-only ecosystem	TensorFlow-centric production systems
		- Data validation, monitoring	- Heavy infrastructure	
		- Airflow/Kubeflow integration		
DVC	**Data versioning & pipelines**	- Git-like data/experiment versioning	- No native model registry/deployment	Teams needing data versioning + pipeline orchestration
		- Pipeline dependency graphs	- Requires external tracking	

Benefits

Best for:

- Teams needing modular tracking and deployment
- Python-heavy ML pipelines
- CI/CD integration with ML models
- Multi-framework projects
- cloud flexibility: Run on local machines, Databricks, Azure ML, or AWS.

Real-World Applications

- **Airbnb:** Uses MLflow to track thousands of model experiments, ensuring reproducibility and allowing teams to compare results across time and teams.
- **Shopify:** Leverages MLflow to handle a high volume of machine learning workflows, with automatic logging and model versioning integrated into their pipelines.
- **Microsoft Azure & AWS:**Support MLflow integration for model tracking and deployment in cloud ML pipelines.

Exemple Workflow in MLflow

- **Track**: Log parameters/metrics during training.

```
import mlflow
mlflow.log_param("n_estimators", 100)
mlflow.log_metric("accuracy", 0.92)
```

- **Package**: Define **MLproject** file for Conda environment.
- **Register**: Add model to registry via UI/API.
-
- **Deploy**: Serve as API using **mlflow models serve**:

  -m runs:/<RUN_ID>/model.

Pros

- Open-source and framework-agnostic
- Integrates easily with cloud and on-premises tools
- Full lifecycle support (track → register → deploy)
- Active community + Databricks backing

Cons

- impler UI compared to competitors
- No native data versioning (combine with DVC if needed)
- Lacks advanced collaboration features (W&B is better here)

Limitations

- No built-in data versioning: Pair with DVC or Delta Lake.
- Limited access control: Use MLflow Server with proxy auth.
- Basic monitoring: Integrate with Prometheus/Grafana.