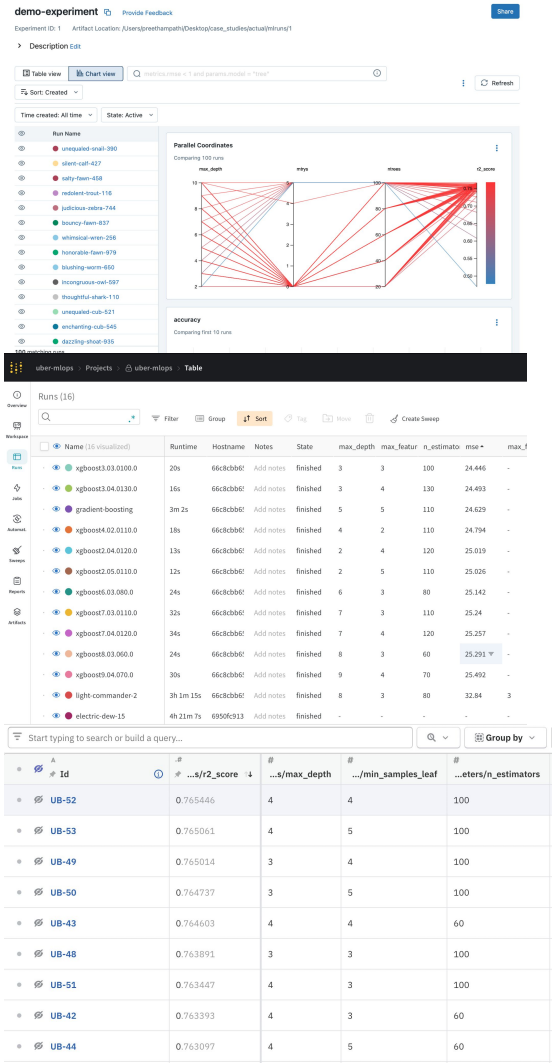
Uber Price Prediction
(Custom Compass)

Akul Bajaj, Preetham Reddy Pathi,
Manish Kumar Vuppugandla, Andres Martinez Tobon

Experiment and Artifact Tracking

Feature	MLFlow	WandB	Neptune
Experiment Tracking	Using with mlflow.start_run() and all runs inside it will be logged in it.	All logs between a start and a finish command will be logged into a single run.	All logs between a start and a end command will be logged into a single run.
Artifact Tracking	Yes	Yes	Yes
Model Registry	Yes	Yes	Yes
Visualisation	Best for Scikit-Learn models	Best for Neural Networks	Yes
Cloud-based	Supports cloud and on-premise deployment	Only cloud-based	Only cloud-based
Pricing	Free and open-source	Free and paid plans available	Free and paid plans available

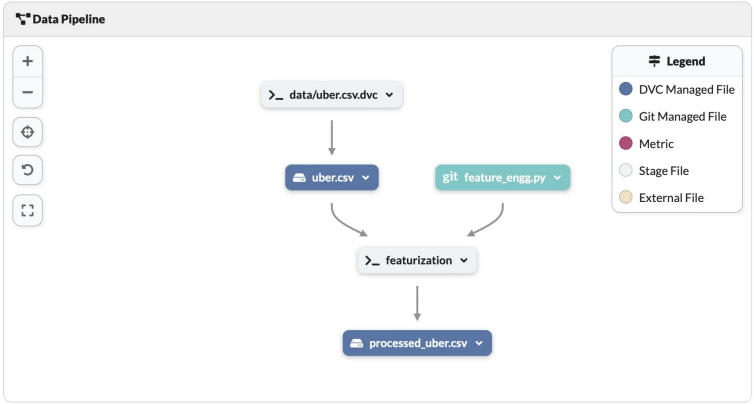


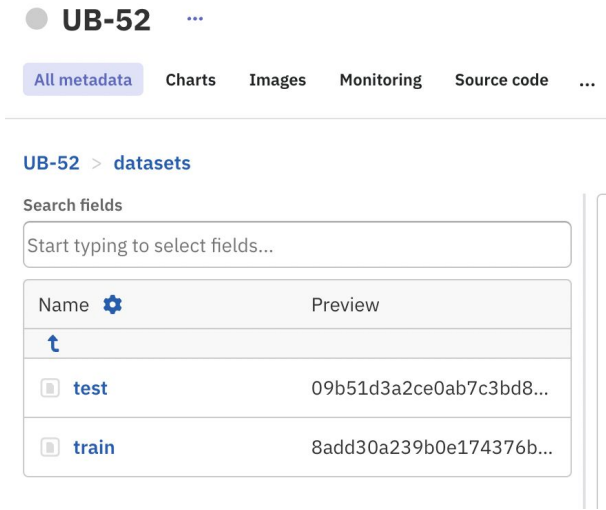
- We're choosing MLFlow for it's superior charts for sklearn models and because it's open-source and free.

Data Versioning

Feature	DVC	Neptune
Data Versioning	Full-fledged	Just saves hash of file
Pipeline Reproducibility	Yes (dvc repro)	No
DagsHub Integration	Yes	No
Cloud-based	Can be used locally or with cloud services	Cloud based
Pricing	Free and open source	Free and paid plans available.

- We chose DVC for our project because of its integration with DagsHub and the single line ‘dvc repro’ pipeline reproducibility.





Data Quality

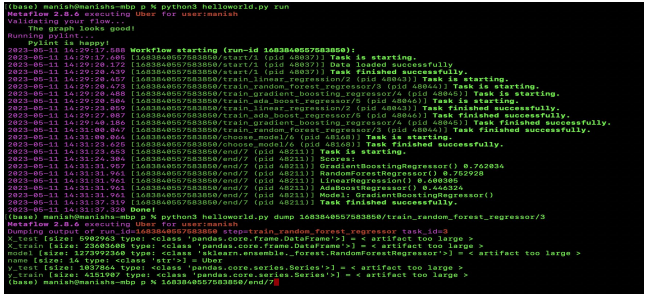
Both Great Expectations and Deepchecks are able to provide data validation checks
Deepchecks stands out because it can be used for model validation as well and can be integrated with other ML tools
We suggest Great Expectations because it's tailored well for our data format and works with our deployment option
Deepchecks would be overkill for our use case

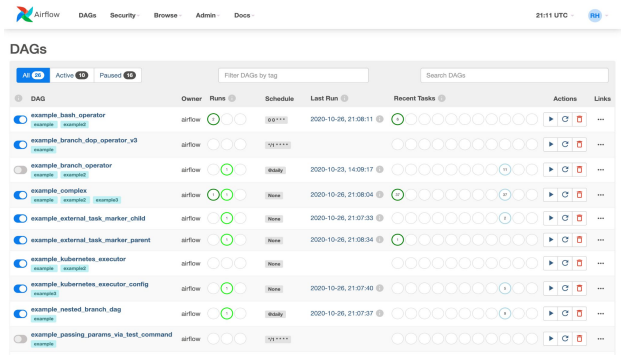
Feature	**Great Expectations**	**Deepchecks**
Key Functionality	• Expectations • Automated data profiling • Data Docs . • Customizable	• Data Integrity • Train-Test Validation • Model Performance Evaluation • Customizable
Integration	• it integrates seamlessly with DAG execution tools like Spark, Airflow, Databricks, etc. • Great Expectations currently works best in a Python environment.	• Spark & Databricks • Airflow • Weights & Biases • HugginFace Transformers • Pytorch
Data Format	• Tabular datasets	• Tabular datasets • Vision Tasks
ML Capabilities	• Provides validation only for data. • Useful at the start of the ML Lifecycle	• Provides validation for both data and models • Offers 3 checks in different phases of the ML flow

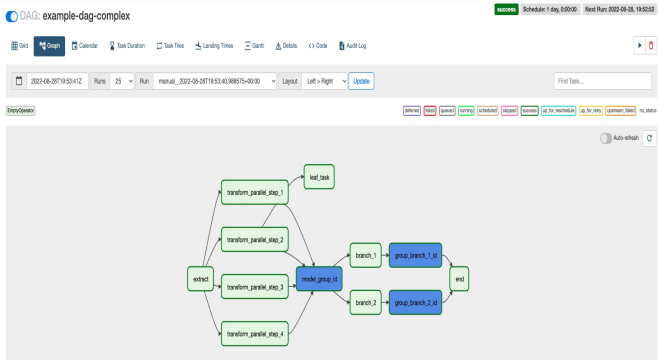
ML Pipeline Orchestration

Feature	*Metaflow*	*Airflow*
Learning curve	Simple, easy, intuitive Python DSL	Steeper learning curve with custom DAGs
In-built server	No (just a CLI)	Built-in UI server
Scalability	Scalable (can handle larger workflows)	Highly scalable (distributed execution & horizontal scaling)
Vendor lock	Strong integration with AWS	No locks, more robust cloud agnostic
Async IO support	Yes, limited (python-async/await)	More robust

- Metaflow is preferred due to its user-friendly and intuitive nature for data scientists.
  (Additionally, as it doesn't require Kubernetes integration, reduces the complexity)





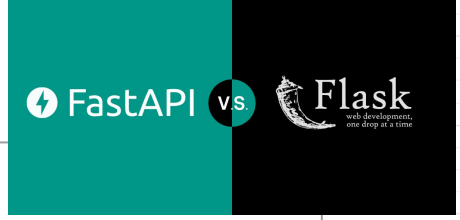


Model Monitoring

Features	***Evidently AI***	***Alibi Detect***
Platform Agnostic	Yes	No (requires Seldon core)
Integration	Easy integration with existing tools such as Metaflow or MLflow	Harder to integrate
Infrastructure Overhead	Low	Requires additional infrastructure
Data Monitoring	Comprehensive coverage of most model monitoring use cases with a simple and clear UI	Specialised outlier, adversarial, and outlier detection methods with support for a wider range of data types
Reporting Capabilities	Limited	Not as comprehensive as Evidently-AI for the full range of data monitoring needs

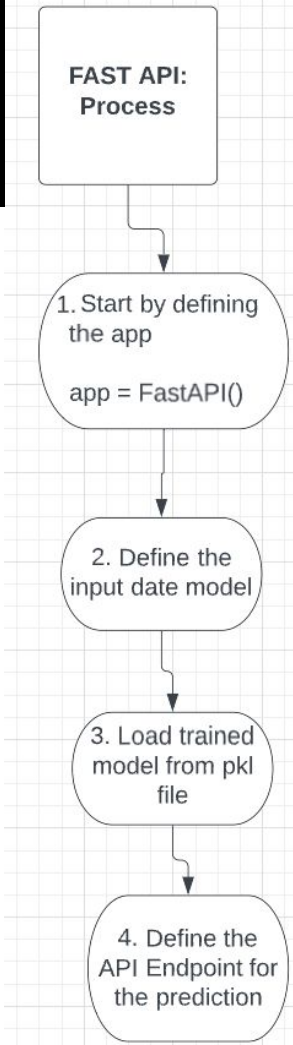
- Evidently-AI is platform agnostic, also offers a simple and clear user interface.

Model Deployment



Feature	Flask	FastAPI
Architecture	Microframework	Modern web framework
Scalability	Good	Good
Ease of use	Very easy	Easy
Testing Support	With libraries	Built-in swagger UI
Inbuilt server	Yes	No (requires uvicorn)
Async Support	No	Yes

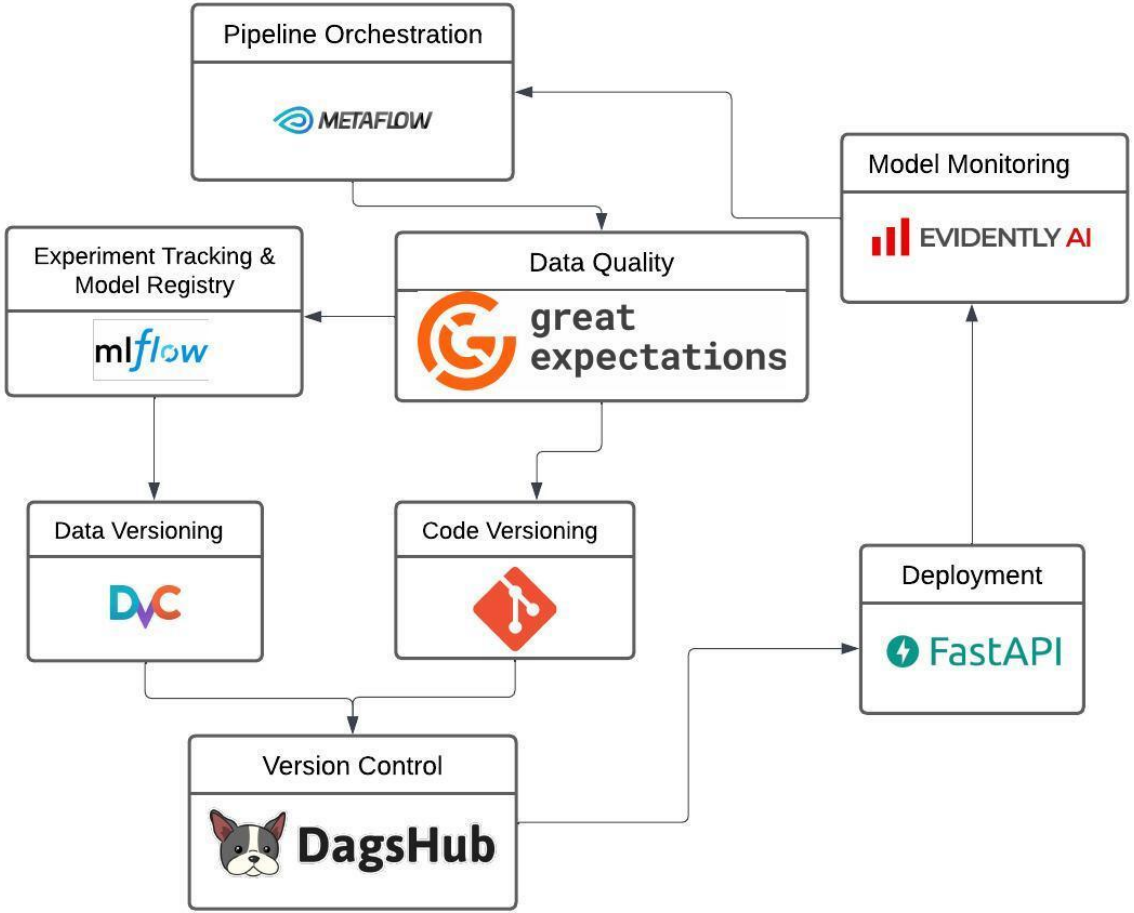
- Although we are not currently using async requests, it will be beneficial to have it for a Machine Learning application. FastAPI makes more sense for our use case.



CI/CD, Linting, Styling, Branching

Feature	**Jenkins**	**GitHub Actions**
Deployment	On-premise or cloud-based	Cloud
Ease of use	Steep learning curve	Easy to use and configure
Integration	Large number of plugins available	Built-in integration with GitHub and other tools.
Cost	Open source and free to use	Free for open-source projects, paid for private.
Scalability	Good	Excellent

Architecture



Thank You