Recap

19 July 2024 16:23

Model Registry

Big picture

DVC pipe

data ver

code

model

exp track

Model Registry

19 July 2024 16:24

$\boxed{\text{model log}}$ × model registry

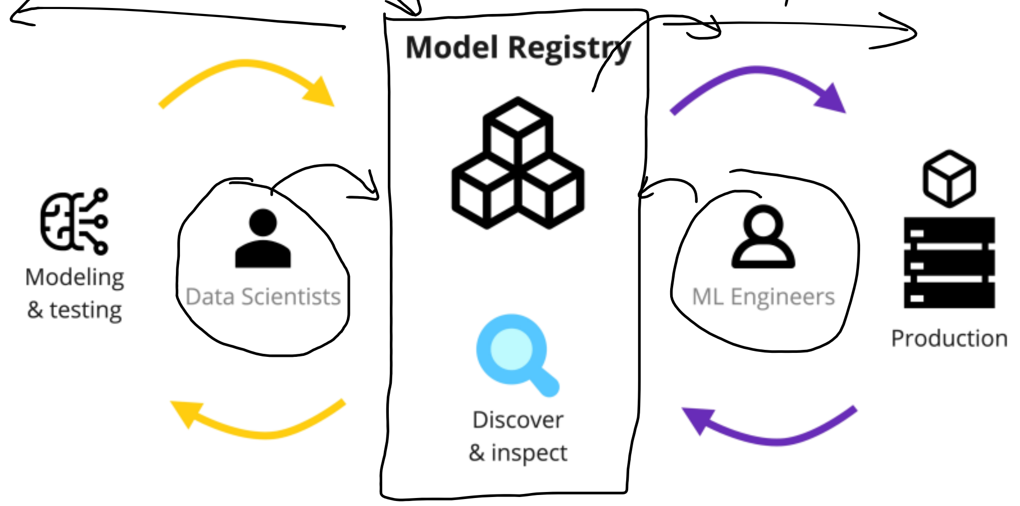
A model registry is a centralized repository that allows data scientists and machine learning engineers to manage the lifecycle of their machine learning models. It provides functionalities for versioning, storing, and organizing models, as well as tracking their metadata and usage history.

Flow

1. Experiments
2. Comparisons
3. Best model
4. Registration
5. Model Serving

$\boxed{\text{repo}}$

ML   mid point   ops



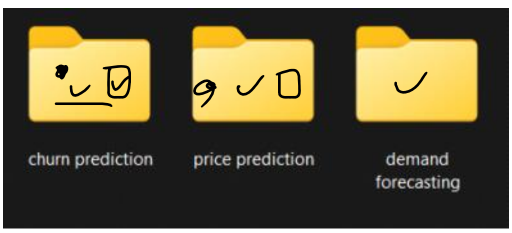
{model registry}   .pkl

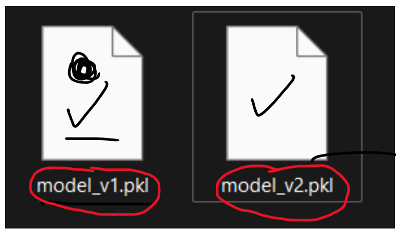
Why do we need a model registry?

Scenario - A data science team working on multiple machine learning problems

v2   swissy   metadata

storage space models





Team	Model name	Parameter	Parameter value	Metric	Metric Value
Churn Prediction	model_v1.pkl	max_depth	5	training_mse	0.19
		n_estimators	100	test_mse	0.23

Core Features

19 July 2024 17:11

1. Model Storage - in files
2. Model Versioning - like model_v2.pkl
3. Metadata Management - parameters stored in a file
4. Model Lineage - Info scattered into multiple files, reproducibility was hard
5. Collaboration - emails
6. Access control & Security
7. Deployment - manually copy pasting
8. Lifecycle Management

ml flow → exp tracking →

model v1

model registry

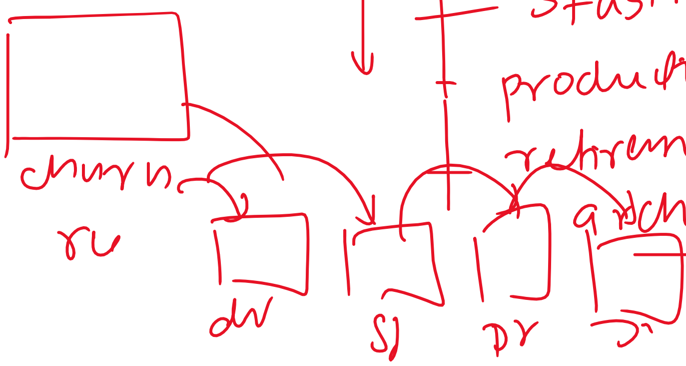
model

4 stages

model

Auditing

- development
- staging
- production
- retirement
- archive — document



Lifecycle Management

19 July 2024 17:14

Stages of a model

1. Development
2. Staging
3. Production
4. Archived

Stage 1 - Development

Objective: The main goal of the development stage is to create and refine the model.

Activities

1. Experiment Tracking
2. Model versioning
3. Artifact storage
4. Metadata logging
5. Evaluation and validation
6. Collaboration and feedback - The registry provides a platform to collaborate, teammates can add comment, tags to the model

Stage 2 - Staging

Objective: The staging stage is used for further testing and validation in an environment that closely resembles production.

dev → None

staging

Activities

- Model Transition - The model, along with its artifacts and metadata, is transitioned from the development stage to the staging stage within the model registry.

```python
mlflow.transition_model_version_stage(
    name="ChurnPredictionModel",
    version=1,
    stage="Staging"
)
```

- Environment Setup - A staging environment is set up that mirrors the production environment as closely as possible. This includes the same software stack, dependencies, and configurations.

- Deploy and Test in Staging environment - The model is deployed to a staging environment, and various tests such as UAT, functional testing, performance testing etc are run to validate its performance.

```
# Example CI/CD pipeline for deploying to staging
steps:
  - name: Deploy model to staging
    script: |
      mlflow models serve -m "models:/ChurnPredictionModel/Staging" -p 5000
  - name: Run tests
```

```
  script: |
    mlflow models serve -m "models:/ChurnPredictionModel/Staging" -p 5000
- name: Run tests
  script: |
    run_tests()  # Custom testing script
```

- Collect & Log feedback
- ```
  mlflow.set_tag("Staging Feedback", "Passed UAT with minor issues")
  ```

Stage 3 - Production

Objective: The production stage is where the model is deployed to serve real-world applications and users.

Activities

- Transition - The model is moved from staging to production

```
mlflow.transition_model_version_stage(
    name="ChurnPredictionModel",
    version=1,
    stage="Production"
)
```

- Deployment - The model is deployed to production environment

```
# Example CI/CD pipeline for deploying to production
steps:
  - name: Deploy model to production
    script: |
      mlflow models serve -m "models:/ChurnPredictionModel/Production" -p 5000
```

- Monitoring - Continuous monitoring of the model's performance is established to track key metrics such as latency, accuracy, and throughput.
- Collecting Feedback - Feedback from end-users is collected and logged in the registry for future improvements

```
mlflow.set_tag("User Feedback", "Positive impact on churn reduction")
```

Stage 4 - Archiving/Retiring

Objective - The archived stage is for models that are no longer in active use but need to be retained for historical, auditing, or regulatory purposes.

Activities

- Assessment - Review model performance and usage metrics to determine if the model should be retired.
- Documentation - Compile and log all relevant documentation in the model registry
```

should be retired.

- Documentation - Compile and log all relevant documentation in the model registry

```
mlflow.set_tag("Documentation", "Complete")
```

- Archiving - Transition the model to the "Archived" stage

```
mlflow.transition_model_version_stage(
    name="ChurnPredictionModel",
    version=1,
    stage="Archived"
)
```

- Deprecation - Update stakeholders and systems to reflect the model's deprecation status

```
mlflow.set_tag("Deprecation Status", "Deprecated")
```

- Compliance and Auditing - Conduct audits and generate compliance reports to ensure all procedures are followed

```
compliance_report = generate_compliance_report()
mlflow.log_artifact(compliance_report)
```

Model Registry Workflow

19 July 2024 17:16

-> run code -> add signature as well
-> register model using UI
-> explore UI -> add description, add tags, trace metadata
-> assign stage
-> change stage

-> register another model
-> change stage

-> register model using code
-> other functionalities using code -> description, tags
-> change state

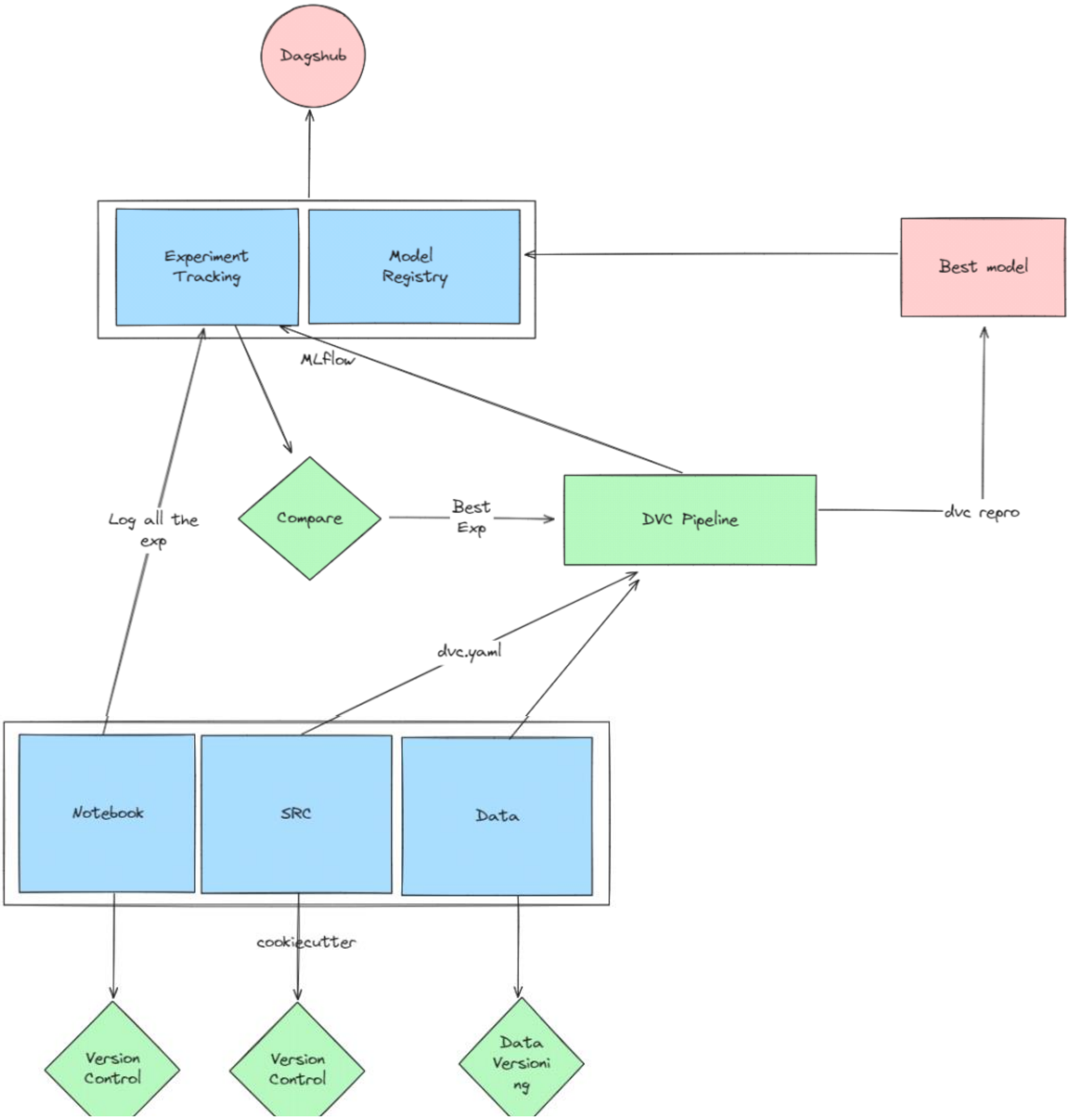
-> model inference

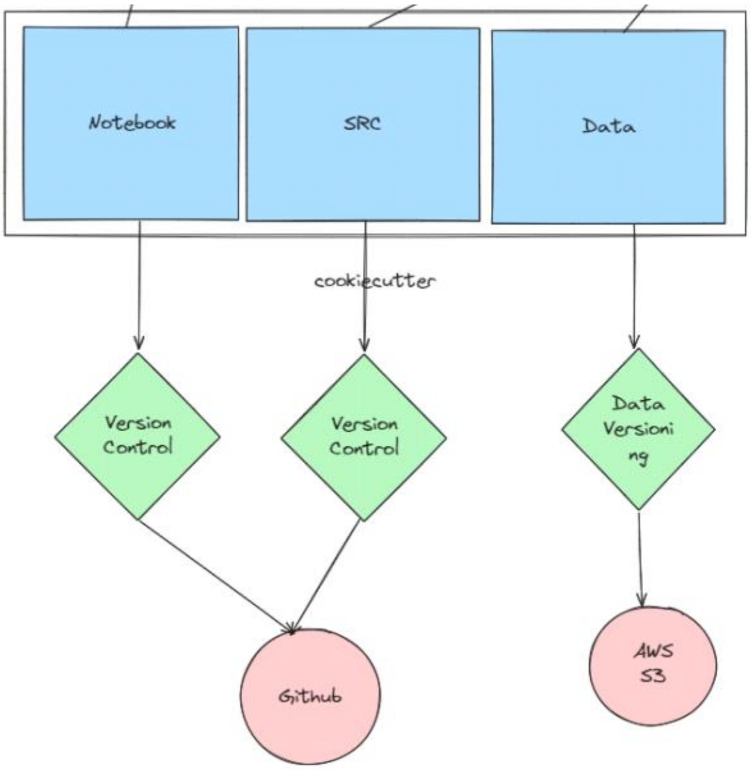
The Big Picture!

19 July 2024 21:42

→ code version
→ pipeline
→ data versioning
→ exp tracking
→ model registry
→ cookie cutter

mlOps




Notebook
SRC
Data
cookiecutter
Version Control
Version Control
Data Versioning
Github
AWS S3