Day71_MLflow_Model_Versioning

August 25, 2025

1 MLflow Model Versioning

In this notebook, we explore how to **version machine learning models** using **MLflow**.
Model versioning is important because it allows us to keep track of different model iterations, compare them, and safely deploy the right version.

We will cover two approaches:

1. **Manual Versioning (UI based):**
   - Log the model during a run.
   - Use the MLflow UI to manually register and promote models.
2. **Automatic Versioning (Code based):**
   - Use `registered_model_name` parameter while logging.
   - MLflow automatically creates new versions for each run.
   - (Optional) Promote versions to *Staging/Production* either via UI or programmatically.

   Note: If you start MLflow with `mlflow ui`, promotion to stages (Staging/Production) must be done manually in the UI.
   For **programmatic stage promotion**, you need to run `mlflow server` with a backend store (e.g., SQLite or MySQL).

2 MLOps: Model Versioning (Manual)

Why model versioning?

- Reproducibility & lineage (know which run made which model)
- Safe rollbacks if a new version fails
- Approvals: move models through None → Staging → Production
- Central catalog for collaboration
- Prepares you for CI/CD automation

2.1 Install dependencies

[ ]:
```
%pip install -q mlflow scikit-learn numpy pandas
```

2.2 Start MLflow Tracking UI in a terminal

Open Anaconda Prompt / Command Prompt, go to your project folder, and run:

```
mlflow ui
```

By default, this will start the tracking server at:

```
http://127.0.0.1:5000
```

Keep this terminal open and running while you use MLflow in Jupyter.

2.3 Configure MLflow tracking & experiment

[1]:
```python
import warnings
warnings.filterwarnings('ignore')
import mlflow, mlflow.sklearn

# Point this to your MLflow Tracking Server
mlflow.set_tracking_uri("http://127.0.0.1:5000")
mlflow.set_experiment("Versioning")

print("Tracking URI:", mlflow.get_tracking_uri())
```

```
Tracking URI: http://127.0.0.1:5000
```

2.4 Prepare dataset

[2]:
```python
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
import pandas as pd

# Synthetic dataset
X, y = make_classification(
    n_samples=1000, n_features=10,
    n_informative=2, n_redundant=8,
    weights=[0.9, 0.1], random_state=42
)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, stratify=y, random_state=42
)

X_train_df = pd.DataFrame(X_train)
print("Train shape:", X_train_df.shape)
```

```
Train shape: (700, 10)
```

2.5 Train & log a model

[3]:
```python
from mlflow.models import infer_signature

params = {"solver": "lbfgs", "max_iter": 1000, "random_state": 42}

with mlflow.start_run() as run:
    model = LogisticRegression(**params)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    report = classification_report(y_test, y_pred, output_dict=True)

    # Log params & metrics
    mlflow.log_params(params)
    mlflow.log_metrics({
        "accuracy": report["accuracy"],
        "f1_macro": report["macro avg"]["f1-score"],
        "recall_1": report["1"]["recall"]
    })

    # Log model with signature
    signature = infer_signature(X_train_df, model.predict(X_train))
    mlflow.sklearn.log_model(
        model,
        artifact_path="model",
        signature=signature,
        input_example=X_train_df.iloc[:5]
    )

    print("Run ID:", run.info.run_id)
    print(" Now open MLflow UI: http://127.0.0.1:5000")
```

```
2025/08/25 10:48:25 WARNING mlflow.models.model: `artifact_path` is deprecated.
Please use `name` instead.
```

```
Downloading artifacts:   0%|          | 0/7 [00:00<?, ?it/s]
```

```
Run ID: 31464588298341f1adc8d0b3b13fdc94
 Now open MLflow UI: http://127.0.0.1:5000
 View run upbeat-ox-946 at: http://127.0.0.1:5000/#/experiments/465653057739855
358/runs/31464588298341f1adc8d0b3b13fdc94
 View experiment at: http://127.0.0.1:5000/#/experiments/465653057739855358
```

2.6 Register manually (UI steps)

- Run in terminal: mlflow ui
- Open: http://127.0.0.1:5000
- Go to: Experiments → Versioning → your run
- Click Artifacts → model → Register model

- Create new model (e.g., “MyRegisteredModel”) → becomes Version 1
- Train again, register again → becomes Version 2

3 MLOps: Model Versioning (Automatic)

3.1 Install dependencies

[ ]:
```
%pip install -q mlflow scikit-learn numpy pandas
```

3.2 Configure MLflow tracking

[4]:
```
import warnings
warnings.filterwarnings('ignore')
import mlflow, mlflow.sklearn

mlflow.set_tracking_uri("http://127.0.0.1:5000")
mlflow.set_experiment("Automatic_Versioning")

print("Tracking URI:", mlflow.get_tracking_uri())
```

```
Tracking URI: http://127.0.0.1:5000
```

3.3 Prepare dataset

[5]:
```
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
import pandas as pd

X, y = make_classification(
    n_samples=1000, n_features=10,
    n_informative=2, n_redundant=8,
    weights=[0.9, 0.1], random_state=42
)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, stratify=y, random_state=42
)

X_train_df = pd.DataFrame(X_train)
print("Train shape:", X_train_df.shape)
```

```
Train shape: (700, 10)
```

3.4 Train, log & auto-register

[6]:
```python
from mlflow.models import infer_signature

REGISTERED_NAME = "MyAutoRegisteredModel"  # change name if you want

params = {"solver": "lbfgs", "max_iter": 1000, "random_state": 42}

with mlflow.start_run() as run:
    model = LogisticRegression(**params)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    report = classification_report(y_test, y_pred, output_dict=True)

    mlflow.log_params(params)
    mlflow.log_metrics({
        "accuracy": report["accuracy"],
        "f1_macro": report["macro avg"]["f1-score"],
        "recall_1": report["1"]["recall"]
    })

    # Auto-register model
    signature = infer_signature(X_train_df, model.predict(X_train))
    mlflow.sklearn.log_model(
        model,
        artifact_path="model",
        signature=signature,
        input_example=X_train_df.iloc[:5],
        registered_model_name=REGISTERED_NAME
    )

    print("Run ID:", run.info.run_id)
    print(" Auto-registered under:", REGISTERED_NAME)
```

```
2025/08/25 10:49:04 WARNING mlflow.models.model: `artifact_path` is deprecated.
Please use `name` instead.

Downloading artifacts:   0%|          | 0/7 [00:00<?, ?it/s]

Registered model 'MyAutoRegisteredModel' already exists. Creating a new version
of this model…
2025/08/25 10:49:20 INFO mlflow.store.model_registry.abstract_store: Waiting up
to 300 seconds for model version to finish creation. Model name:
MyAutoRegisteredModel, version 4
Created version '4' of model 'MyAutoRegisteredModel'.

Run ID: a616d95e48c449c196098398870950a9
 Auto-registered under: MyAutoRegisteredModel
 View run likeable-robin-144 at: http://127.0.0.1:5000/#/experiments/6361712600
90459286/runs/a616d95e48c449c196098398870950a9
```

View experiment at: http://127.0.0.1:5000/#/experiments/636171260090459286

3.5 Check & promote programmatically

[ ]:
```python
from mlflow.tracking import MlflowClient

client = MlflowClient()
versions = client.search_model_versions(f"name='{REGISTERED_NAME}'")

for mv in versions:
    print("Name:", mv.name, "| Version:", mv.version, "| Stage:", mv.
 ↪current_stage)

# Promote latest version to Staging
if versions:
    latest_version = max(int(mv.version) for mv in versions)
    client.transition_model_version_stage(
        name=REGISTERED_NAME,
        version=str(latest_version),
        stage="Staging",
        archive_existing_versions=True
    )
    print(" Promoted version", latest_version, "to Staging")
```

```
Name: MyAutoRegisteredModel | Version: 5 | Stage: None
Name: MyAutoRegisteredModel | Version: 4 | Stage: None
Name: MyAutoRegisteredModel | Version: 2 | Stage: Archived
Name: MyAutoRegisteredModel | Version: 3 | Stage: None
Name: MyAutoRegisteredModel | Version: 1 | Stage: None
```

- First run → Version 1
- Second run (this code) → Version 2
- MLflow UI → check Models → MyAutoRegisteredModel
- You'll see two versions listed

3.6 Code to Train Again (Version 2)

[7]:
```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from mlflow.models import infer_signature
import mlflow

REGISTERED_NAME = "MyAutoRegisteredModel"  # same name as before

params = {"solver": "lbfgs", "max_iter": 2000, "random_state": 99}  # slight␣
 ↪change in params

with mlflow.start_run() as run:
```

```
    model = LogisticRegression(**params)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    report = classification_report(y_test, y_pred, output_dict=True)

    # Log params & metrics
    mlflow.log_params(params)
    mlflow.log_metrics({
        "accuracy": report["accuracy"],
        "f1_macro": report["macro avg"]["f1-score"],
        "recall_1": report["1"]["recall"]
    })

    # Auto-register as new version
    signature = infer_signature(X_train_df, model.predict(X_train))
    mlflow.sklearn.log_model(
        model,
        name="model",    # new MLflow syntax
        signature=signature,
        input_example=X_train_df.iloc[:5],
        registered_model_name=REGISTERED_NAME
    )

    print("Run ID:", run.info.run_id)
    print(" Auto-registered new version under:", REGISTERED_NAME)
```

```
Downloading artifacts:   0%|          | 0/7 [00:00<?, ?it/s]
```

```
Registered model 'MyAutoRegisteredModel' already exists. Creating a new version
of this model…
2025/08/25 10:50:14 INFO mlflow.store.model_registry.abstract_store: Waiting up
to 300 seconds for model version to finish creation. Model name:
MyAutoRegisteredModel, version 5
Created version '5' of model 'MyAutoRegisteredModel'.
```

```
Run ID: 618ebc42c4e9433fbb1ebb54ad3b2f79
 Auto-registered new version under: MyAutoRegisteredModel
 View run bemused-gull-614 at: http://127.0.0.1:5000/#/experiments/636171260090
459286/runs/618ebc42c4e9433fbb1ebb54ad3b2f79
 View experiment at: http://127.0.0.1:5000/#/experiments/636171260090459286
```

3.7 Check Versions & Promote

[ ]:
```
from mlflow.tracking import MlflowClient

client = MlflowClient()
versions = client.search_model_versions(f"name='{REGISTERED_NAME}'")

for mv in versions:
```

```
    print("Name:", mv.name, "| Version:", mv.version, "| Stage:", mv.current_stage)

# Promote latest version to Staging
if versions:
    latest_version = max(int(mv.version) for mv in versions)
    client.transition_model_version_stage(
        name=REGISTERED_NAME,
        version=str(latest_version),
        stage="Staging",
        archive_existing_versions=True
    )
    print(" Promoted version", latest_version, "to Staging")
```

4 MLflow Model Versioning

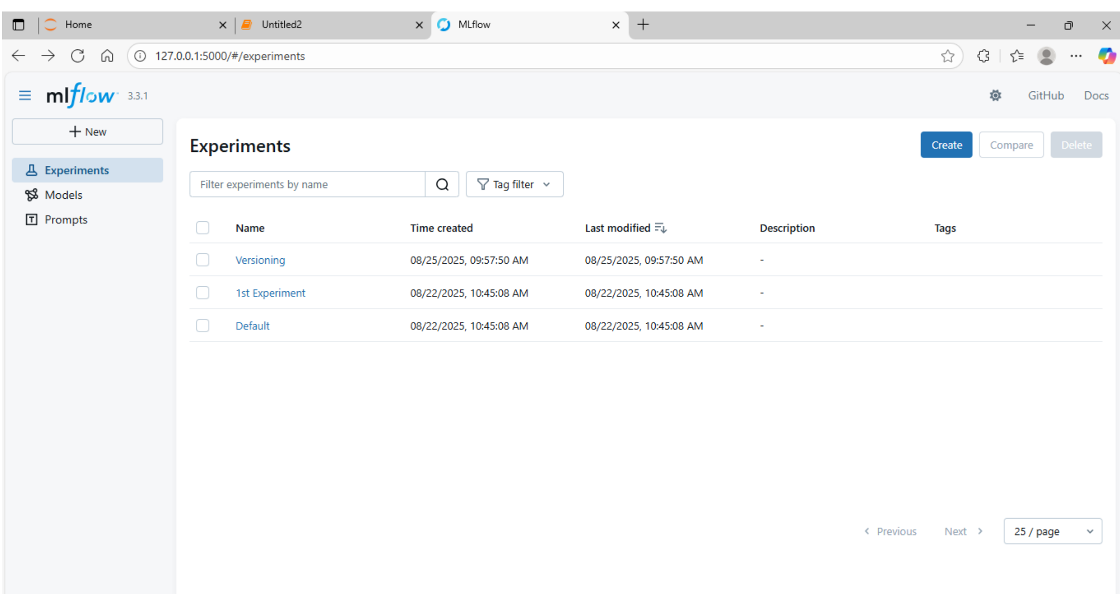
In MLflow, we can version models in two ways:

1. **Manual Versioning (UI based)**
2. **Automatic Versioning (Code based with `registered_model_name`)**

4.1 Manual Model Versioning

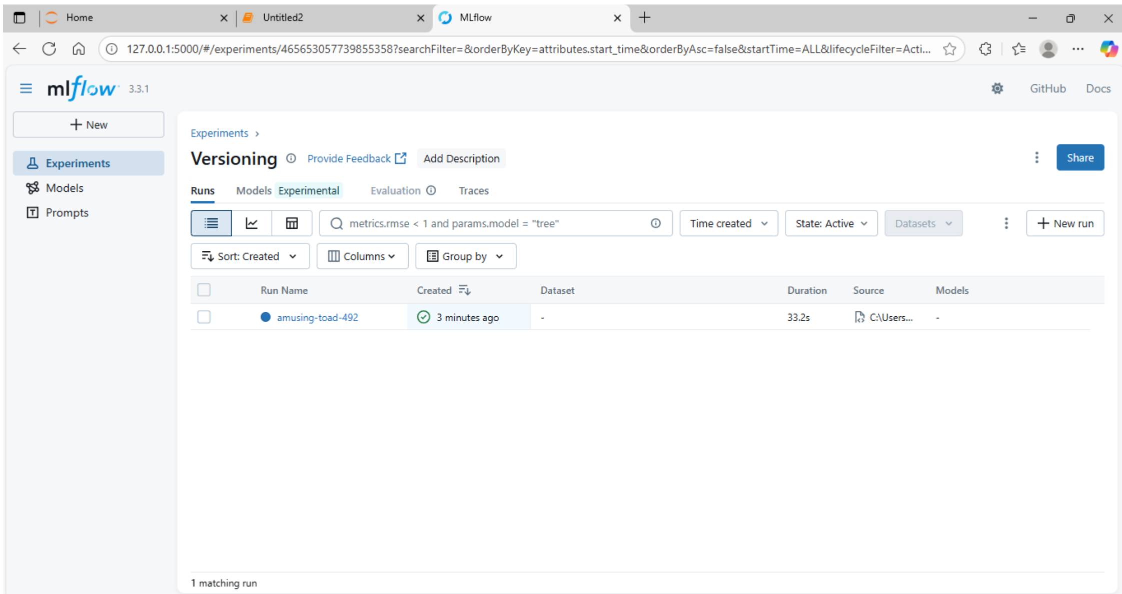
In manual versioning, we first log a model inside an experiment run, and then **manually register** it as a version in the UI.

Step 1: Experiments Page (o1)



Here we can see all experiments created. In this case, the experiment `Versioning` was used for manual registration.

Step 2: Run Details → Register Model (o2)



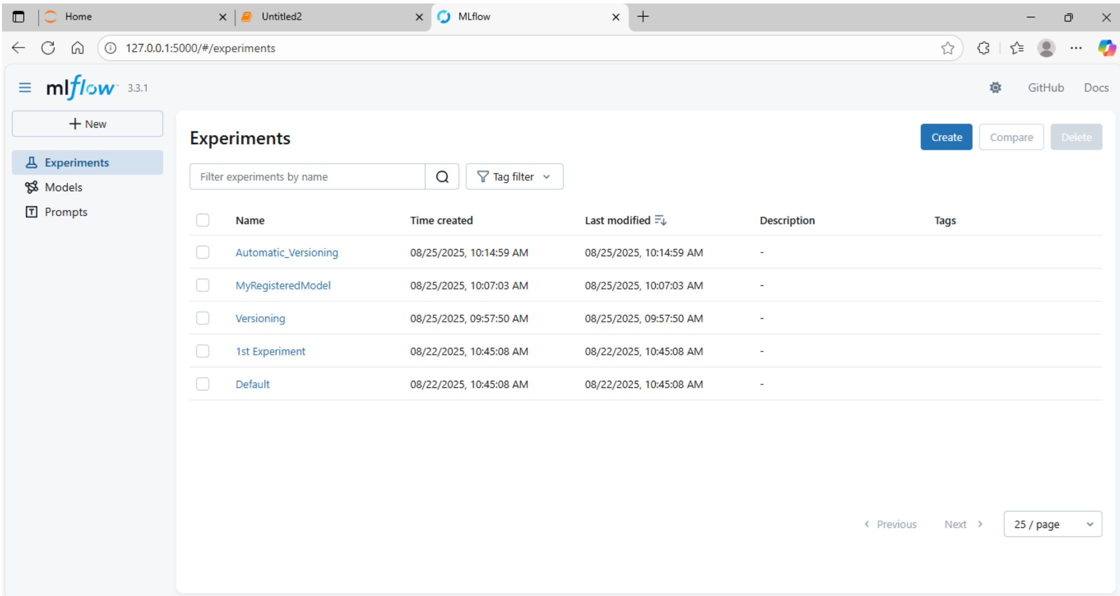
Inside the `Versioning` experiment, a run was executed. From the run details, under **Artifacts → model**, we click **Register Model**.
- If the model is new → create a new model name (`MyRegisteredModel`).
- If the model already exists → it will create **Version 2, 3, ...** under that model.

This way we manually control model versions from the UI.

4.2 2. Automatic Model Versioning

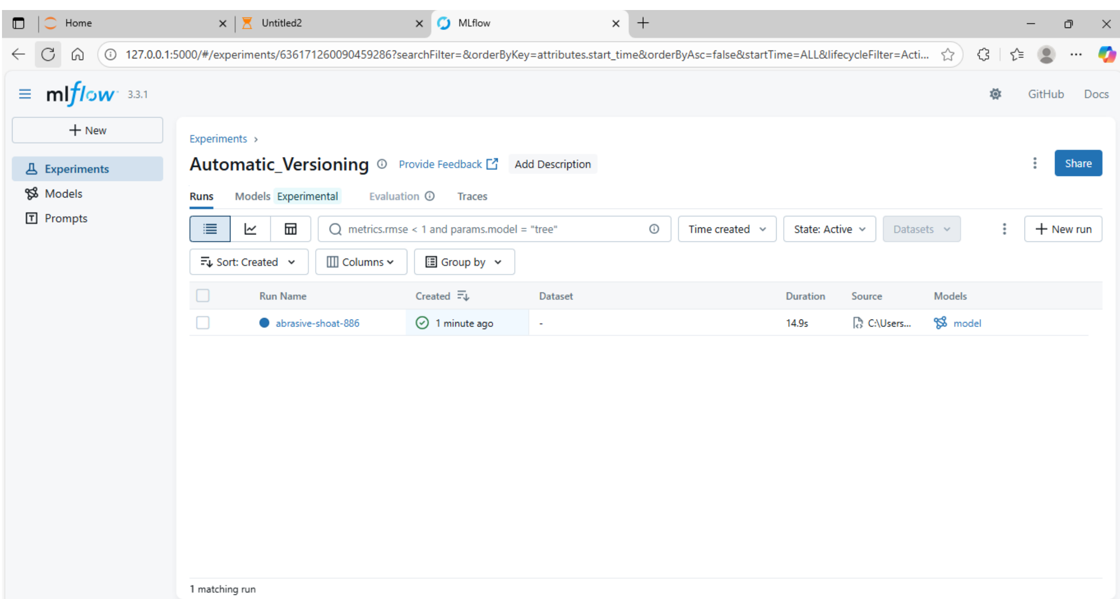
Automatic versioning happens directly from code when we pass `registered_model_name` while logging the model.
Every run automatically creates the next version.

Step 1: Experiments Page with Auto-Versioning (o3)



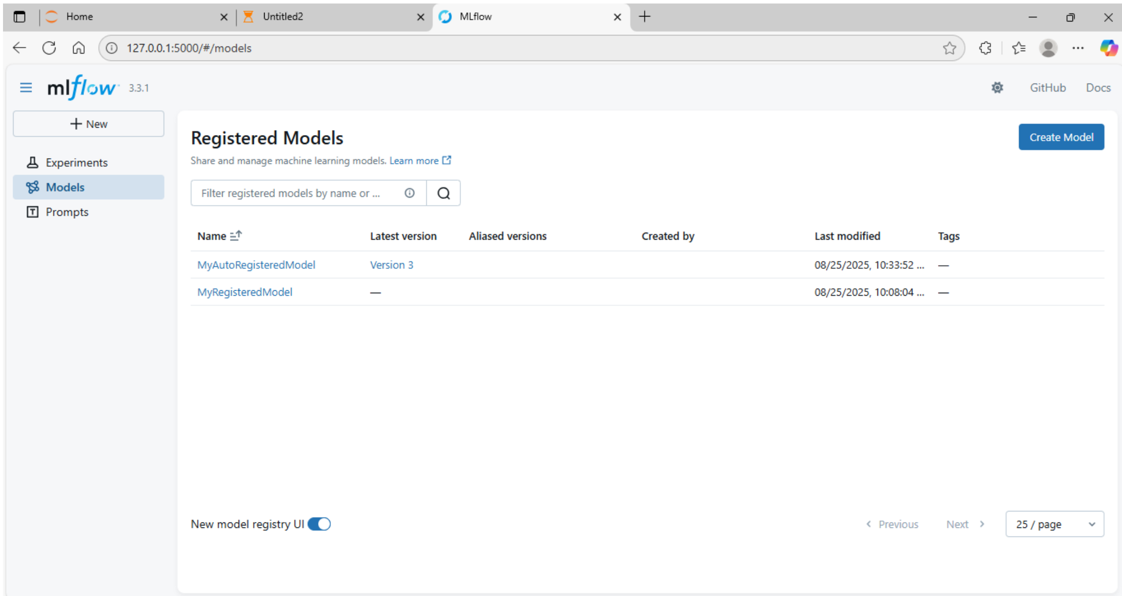
Here a new experiment `Automatic_Versioning` was created when running code with `registered_model_name="MyAutoRegisteredModel"`.

Step 2: Run Logs a Model (o4)



In the run details, you can see the **Models column** already has the model logged. This is because the model was auto-registered in the registry without manual UI clicks.

Step 3: Model Registry with Multiple Versions (o5)



The **Model Registry** shows the registered model `MyAutoRegisteredModel`. Each time we re-run the code, a new version is created automatically (Version 1, 2, 3, ...).
From here we can also promote versions to **Staging** or **Production**.

Summary

- **Manual Versioning**: Log the model → go to UI → register model manually → creates Version 1, 2, ...

- **Automatic Versioning**: Use `registered_model_name` in code → each run automatically creates the next version in the registry.

5 Conclusion

In this notebook, we explored **MLflow Model Versioning** using two approaches:

1. **Manual Versioning (UI-based)** – registering models directly from the MLflow UI.

2. **Automatic Versioning (Code-based)** – registering models automatically using `registered_model_name` in the code.

We also learned how to track experiments, register models, and manage different **versions (v1, v2, ...)** in the **Model Registry**.
Finally, we saw how models can be promoted to stages like **Staging** or **Production**, which is an essential part of the **MLOps lifecycle**.

This sets the foundation for more advanced MLOps tasks like **CI/CD pipelines, deployment, and monitoring**.