MATRYCS: MLFlow tutorial

Gregor Cerar, PhD
Comsensus

MATRYCS

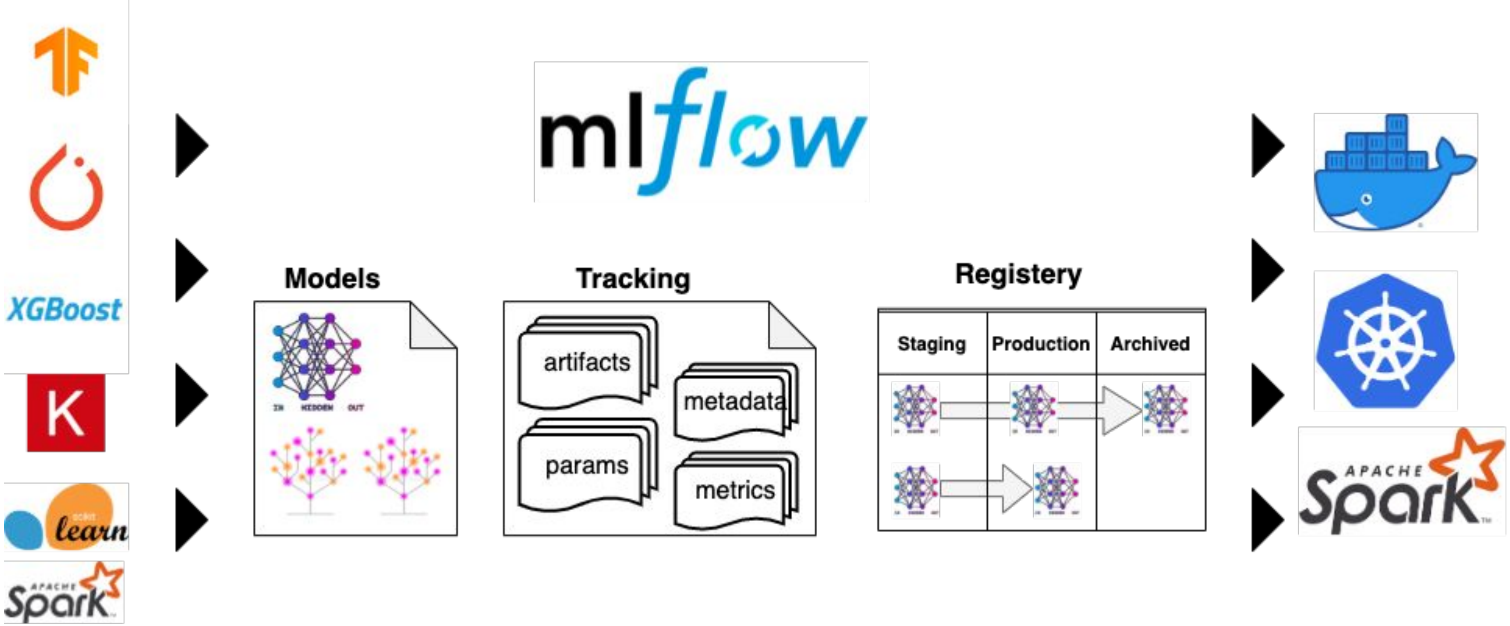
What MFlow IS?

- Model development aspect:
    - ... is a service to **store** binary blobs of the models.
    - ... is a service to **track** changes and progress of ML development.
    - ... is a service that helps propagate information about which models are production-ready. Models are cherry-picked by a leading model developer or automated through an external service/script.
- Model deployment aspect:
    - ... is a service that provides labels "production," and "staging" to help determine which models are suitable for deployment. ***Hint:*** More recent is not often better.
    - ... can serve models on its own

What MLFlow IS NOT?

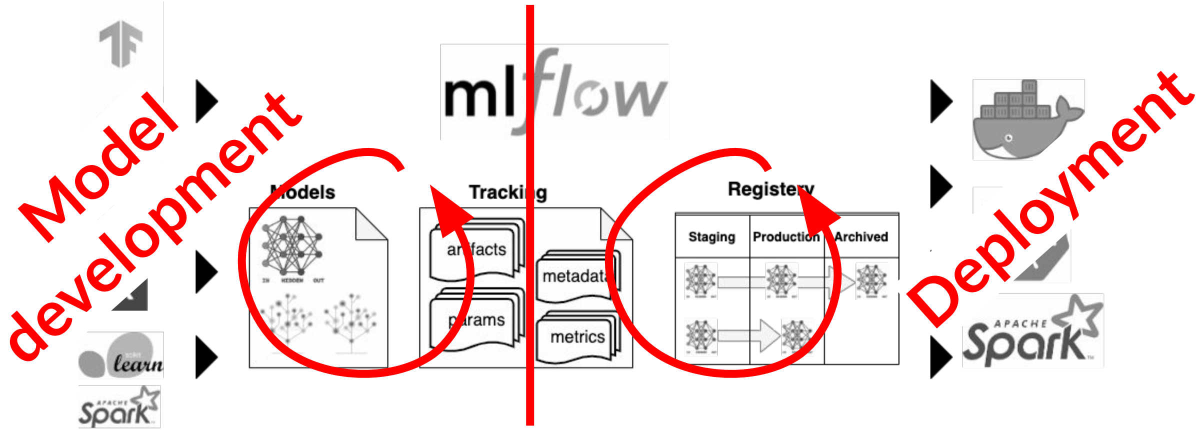
- ... a Continuous retraining system. **Why?** MLFlow has no clue what data was used from pushed models nor where data came from. Retraining is typically done by, for instance, ArgoCD, Flux, GitLab, DataBricks, or KubeFlow.
- ... a MLOps management system. **Why?** MLFlow is simple & minimalistic, but it has no security integrated. No permissions. No authentication. That's why it is not exposed to the Internet through public IP. Its web UI is not meant for a production environment. It is not a good idea to have it accessible to everyone. **Inside MATRYCS project, service is secured with KeyCloak.**

The purpose of MLFlow (1/2)



The purpose of MLFlow (2/2)

- Keep the records of models, and track the progress of ML model development. This functionality is referred to as a "model registry."
- Decouples model development (data science) and model deployment (DevOps).

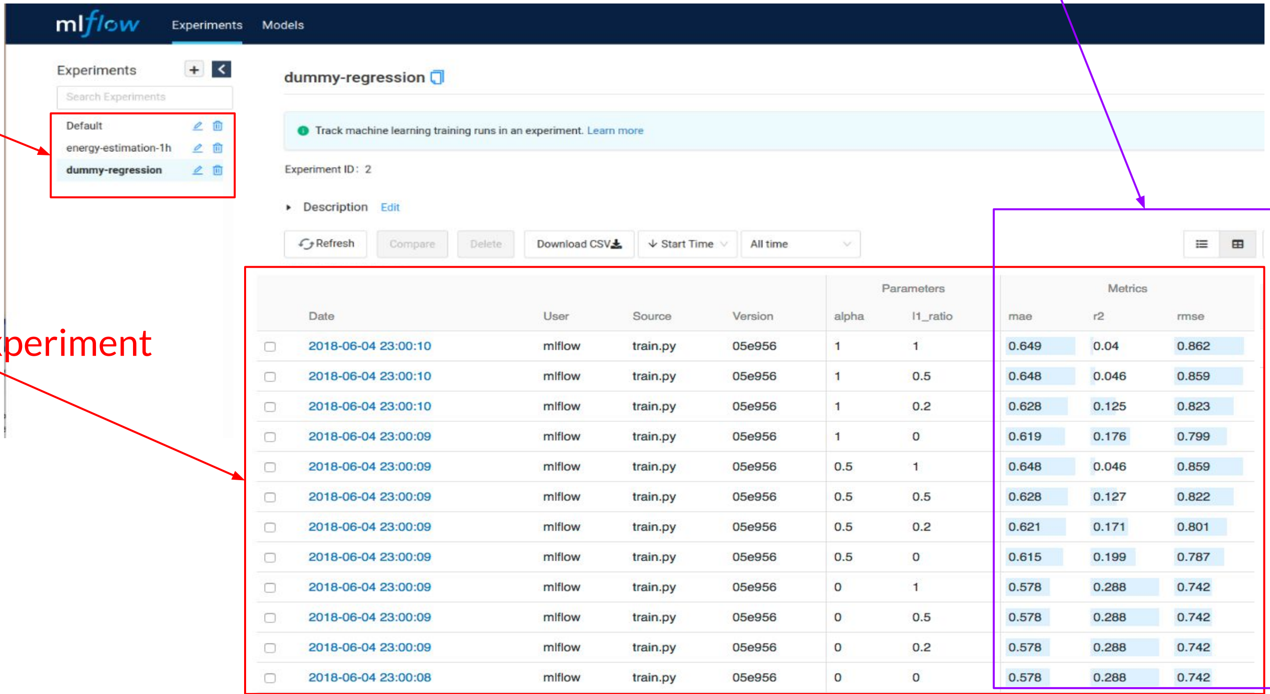


MLFlow terminology (1/2)

Performance metrics of each attempt

Experiment(s)

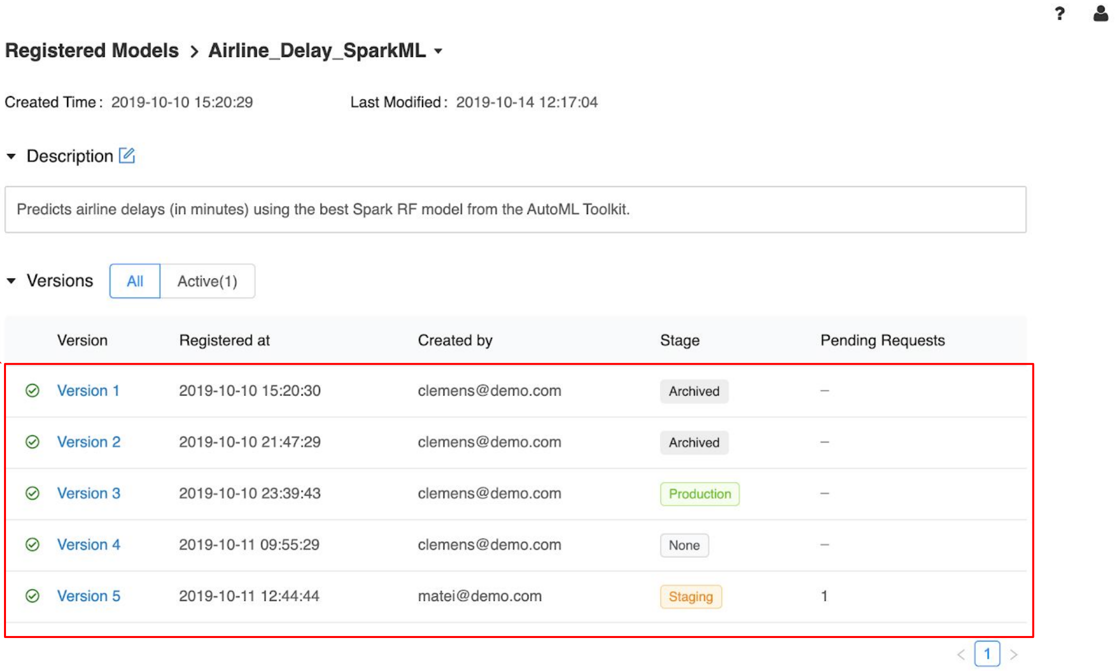
Summary of attempts in experiment



				Parameters		Metrics		
Date	User	Source	Version	alpha	l1_ratio	mae	r2	rmse
2018-06-04 23:00:10	mlflow	train.py	05e956	1	1	0.649	0.04	0.862
2018-06-04 23:00:10	mlflow	train.py	05e956	1	0.5	0.648	0.046	0.859
2018-06-04 23:00:10	mlflow	train.py	05e956	1	0.2	0.628	0.125	0.823
2018-06-04 23:00:09	mlflow	train.py	05e956	1	0	0.619	0.176	0.799
2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	1	0.648	0.046	0.859
2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	0.5	0.628	0.127	0.822
2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	0.2	0.621	0.171	0.801
2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	0	0.615	0.199	0.787
2018-06-04 23:00:09	mlflow	train.py	05e956	0	1	0.578	0.288	0.742
2018-06-04 23:00:09	mlflow	train.py	05e956	0	0.5	0.578	0.288	0.742
2018-06-04 23:00:09	mlflow	train.py	05e956	0	0.2	0.578	0.288	0.742
2018-06-04 23:00:08	mlflow	train.py	05e956	0	0	0.578	0.288	0.742

MLFlow terminology (2/2)

Versions of the
registered model
and its status



What are model development team responsibilities?

- Add extra lines of code to push models to MLFlow.
- Integrate with Apache Airflow for retraining. Why? MLFlow has no idea where data comes from. That is the responsibility of the ML model developer.
- Update "staging" and "production" labels as needed. The model development team is responsible for cherry-picking "the right" model for production.

Push built ML model to MLFlow

Pre-requirements:

- Working script that builds the ML model.
- MFLow experiment name already exists

Steps:

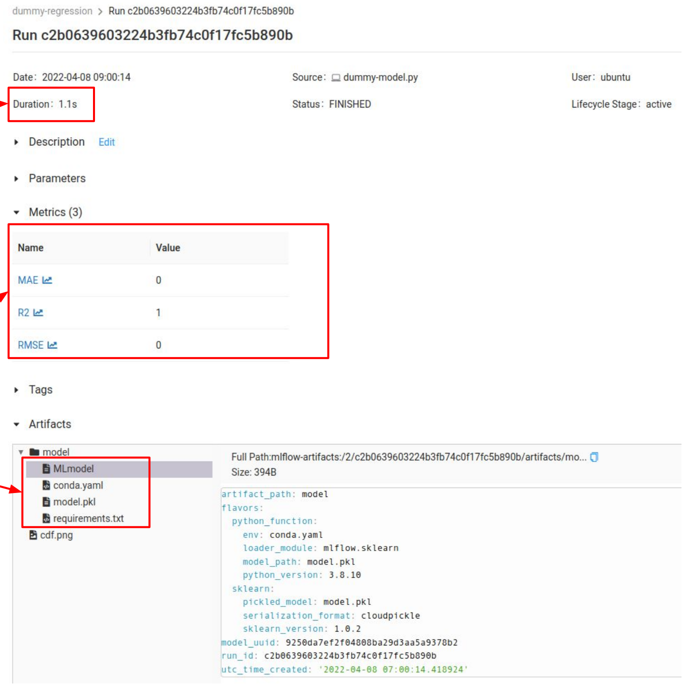
1. Install MLFlow as dependency to your project
2. Import ***mlflow*** package
3. Set URI to MLFlow server
4. Set MLFlow experiment name
5. Call MLFlow method to store model
6. Run the script & model will appear on server

```python
class config:
    SEED = 42
    MLFLOW_TRACKING_URI = 'http://192.168.0.76:5000/'
    EXPERIMENT_NAME = 'dummy-regression'
    REGISTERED_MODEL_NAME = 'matrycs-dummy-regressor'


# Configure logger
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

# Configure MLFLow settings to be aware of remote tracker server
mlflow.set_tracking_uri(config.MLFLOW_TRACKING_URI)
mlflow.set_experiment(config.EXPERIMENT_NAME)
```

```python
    # MLFlow will store model into pickle for us.
    mlflow.sklearn.log_model(
        sk_model=model,
        artifact_path='model',
        registered_model_name=config.REGISTERED_MODEL_NAME,
        pip_requirements=['-r ./requirements.txt'],
    )
```

Example: Push attempt to experiment

```python
model = dummy.DummyRegressor(strategy='constant', constant=1)

# .start_run() and .end_run() measure time to fit the data.
# This is more in case of deep learning.
mlflow.start_run()
model.fit(X_train, y_train)
mlflow.end_run()

# predict values for evaluation
y_pred = model.predict(X_test)

# MLFlow will store model into pickle for us.
mlflow.sklearn.log_model(
    sk_model=model,
    artifact_path='model',
    registered_model_name=config.REGISTERED_MODEL_NAME, # model registration
    pip_requirements=['-r ./requirements.txt'],
)

# Log all relevant metrics for given task
rmse = metrics.mean_squared_error(y_test, y_pred, squared=False)
mlflow.log_metric('RMSE', rmse)

mae = metrics.mean_absolute_error(y_test, y_pred)
mlflow.log_metric('MAE', mae)

r2 = metrics.r2_score(y_test, y_pred)
mlflow.log_metric('R2', r2)
```



What are model deployment team responsibilities?

- Integration with Apache Airflow for awareness of when to re-deploy. An alternative is polling the MLFlow server for updates.
- Deploy only models labeled by the model development team.

Example: Programmatically list all registered models

Create client instance

Access all registered models

Access latest versions of model

```python
def main():
    """Let's retrieve and print some information from MLFlow."""

    client = mlflow.tracking.MlflowClient(
        tracking_uri=config.MLFLOW_TRACKING_URI,
    )

    # Print every registered model
    print('\n\nView all registered models:')
    for mv in client.list_registered_models():
        pprint(dict(mv), indent=4)

    # Retrieve latest releases, if we know registered name.
    print(f'\n\nView latest versions of "{config.REGISTERED_MODEL_NAME}" model:')
    for mv in client.get_latest_versions(name=config.REGISTERED_MODEL_NAME):
        pprint(dict(mv), indent=4)


if __name__ == "__main__":
    main()
```

Example: Programmatically pull latest model

URI to MLFlow service

Obtain model (*.pkl) from MLFlow with helper function

Let model predict value

```python
class config:
    MLFLOW_TRACKING_URI = 'http://192.168.0.76:5000/'
    EXPERIMENT_NAME = 'dummy-regression'
    REGISTERED_MODEL_NAME = 'matrycs-dummy-regressor'

    STAGE = 'Production'

      You, 9 hours ago • Initial commit …
# Configure logger
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

# Configure MLFLow settings to be aware of remote tracker server
mlflow.set_tracking_uri(config.MLFLOW_TRACKING_URI)

def main():
    model = mlflow.pyfunc.load_model(
        model_uri=f'models:/{config.REGISTERED_MODEL_NAME}/{config.STAGE}',
        suppress_warnings=False,
    )

    # Retrieve the data
    N_SAMPLES = 2
    N_FEATURES = 4  # 'matrycs-dummy-regressor' accepts 4 inputs

    input_size = (N_SAMPLES, N_FEATURES)
    inputs = np.random.random(input_size)

    outputs = model.predict(inputs)
    print(f'Model output for {N_SAMPLES} sample(s):\n\t{outputs}')


if __name__ == "__main__":
    main()
```

What is the end goal?

- Fully automated MLOps pipeline, which MLFlow is part of.
    1. Apache Airflow triggers retraining.
    2. Python scripts (or Jupyter notebooks) produces newer model on new data.
    3. Python scripts (or Notebook) push a new binary model to MLFlow.
    4. (optionally) If a new model is tagged as "production," Apache Airflow (or polling service) would trigger an update of a deployment.

Resources

MLFlow documentation: https://www.mlflow.org/docs/latest/model-registry.html

MLFlow deployment & examples: https://github.com/MATRYCS/ml_model_tracking_framework

How to get access to MLFlow deployment?

For local & test deployment try our docker-compose based solution. Instructions:

- https://github.com/MATRYCS/ml_model_tracking_framework/tree/main/mlflow

MATRYCS project partners use deployment accessible from EGI's internal network using KeyCloak

- IPv4 address: 192.168.0.76
- Within testing period, instance can be accessed through VPN or SSH port forwarding.
  - Port forwarding: *ssh -L 5000:192.168.0.76:5000 <MATRYCS-SSH-PROXY-IPv4-ADDR>*
  - After port forwarding MLFlow is accessible through localhost: *http://127.0.0.1:5000*

Contact

Gregor Cerar, Comsensus, Slovenia

gregor.cerar@comsensus.si