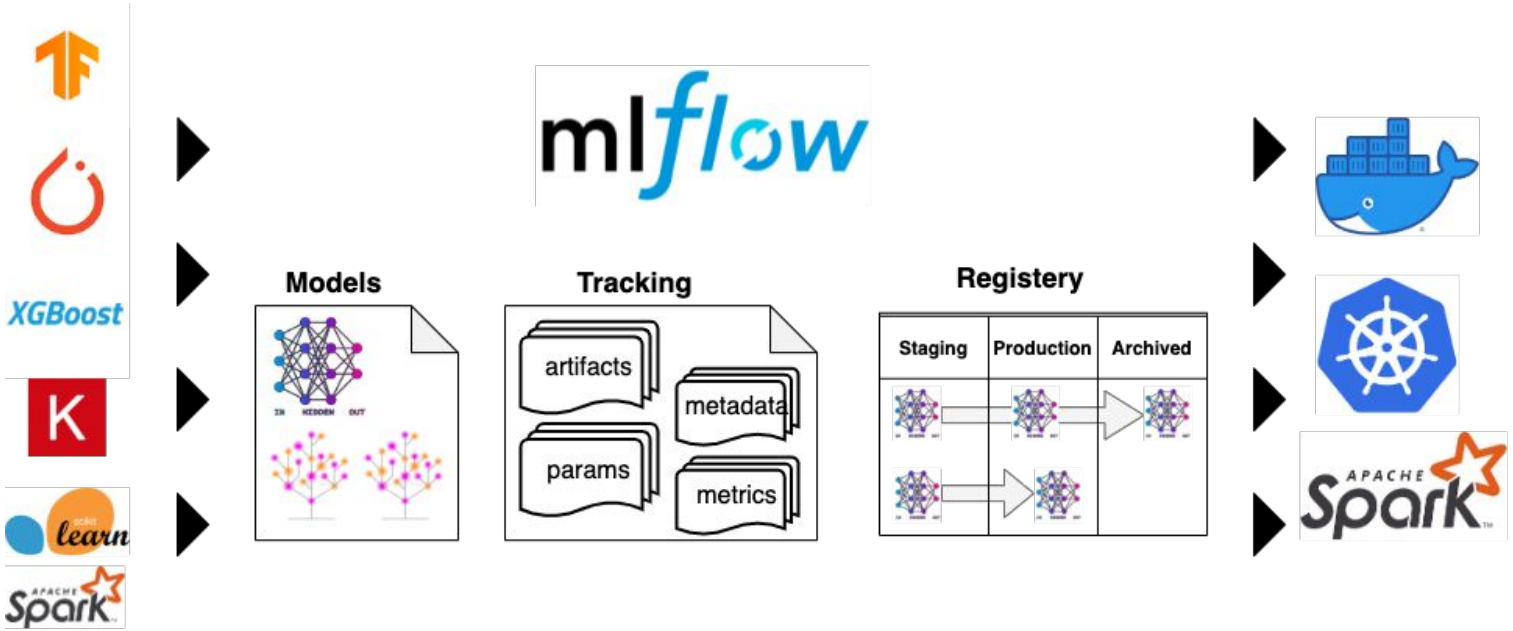
MATRYCS: MLFlow tutorial

Gregor Cerar, PhD
Comsensus

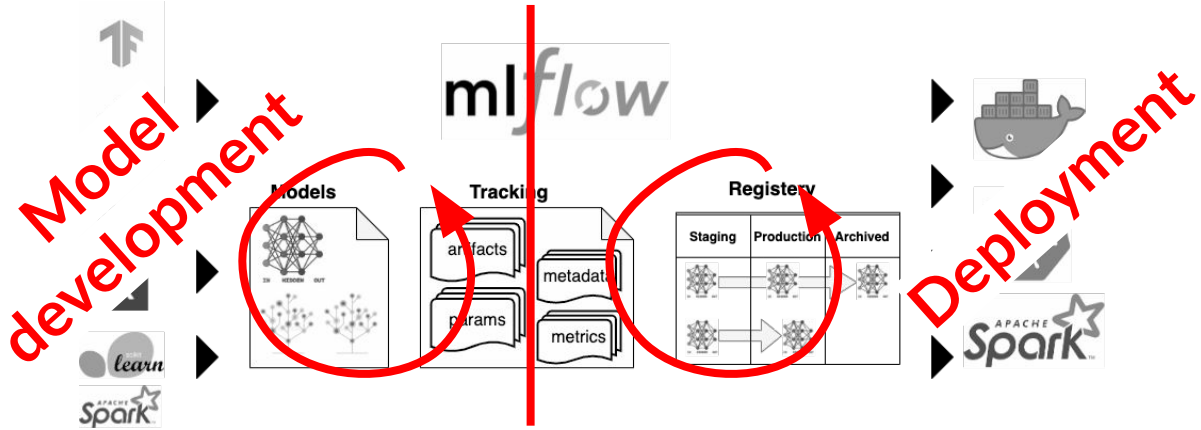
MATRYCS

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).

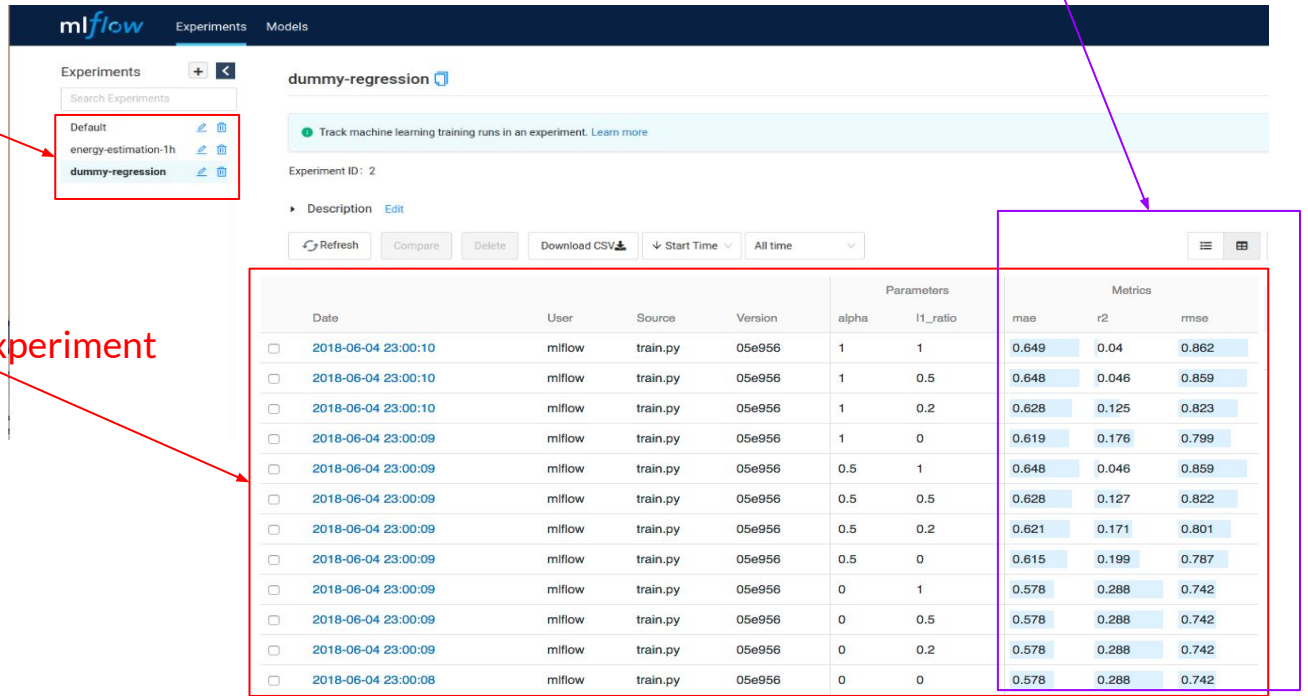


MLFlow terminology (1/2)

Performance metrics of each attempt

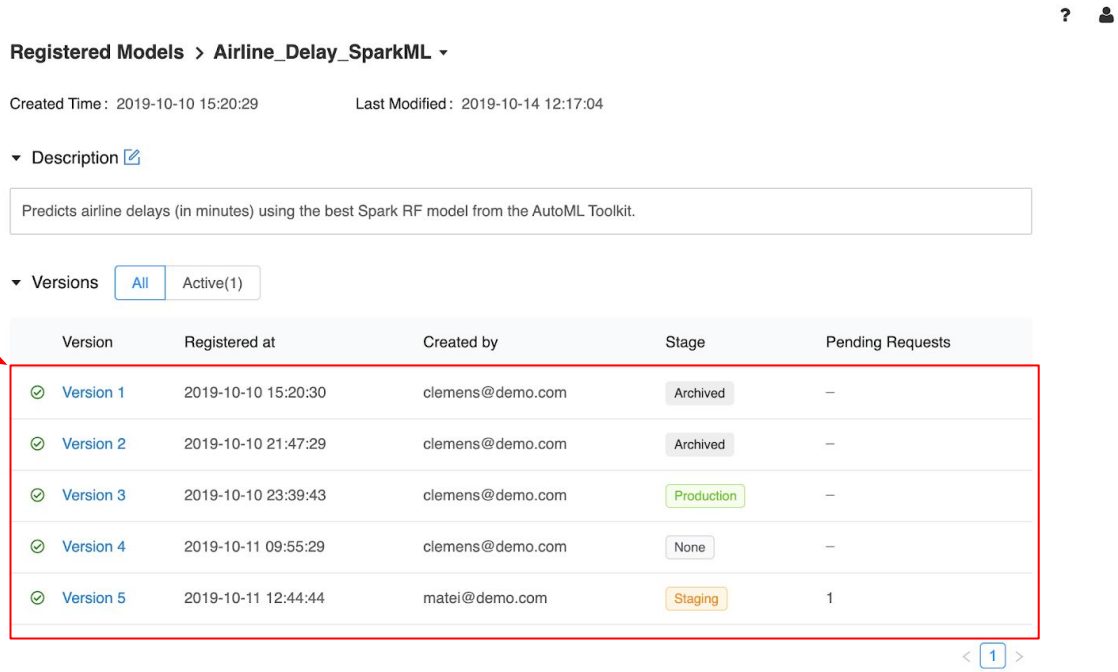
Experiment(s)

Summary of attempts in experiment



MLFlow terminology (2/2)

Versions of the registered model and its status



Push built ML model to MLFlow

Pre-requirements:

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

Steps:

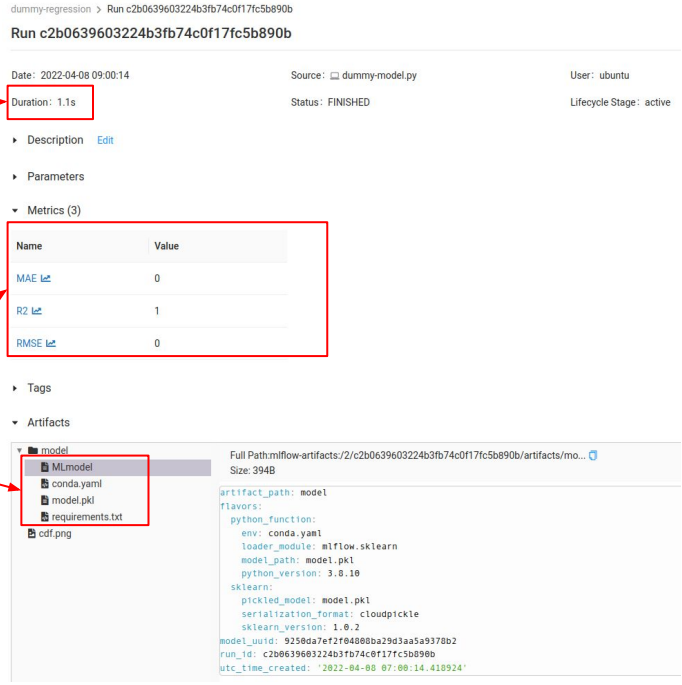
1. Install MLFlow as dependency to your project
2. Import ***mlflow*** package
3. Set URI to MLFlow server
4. Set MLFlow experiment name
5. Call MLFlow method to store model
6. Run the script & model will appear on server

```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)
```



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()
```

Resources

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

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