



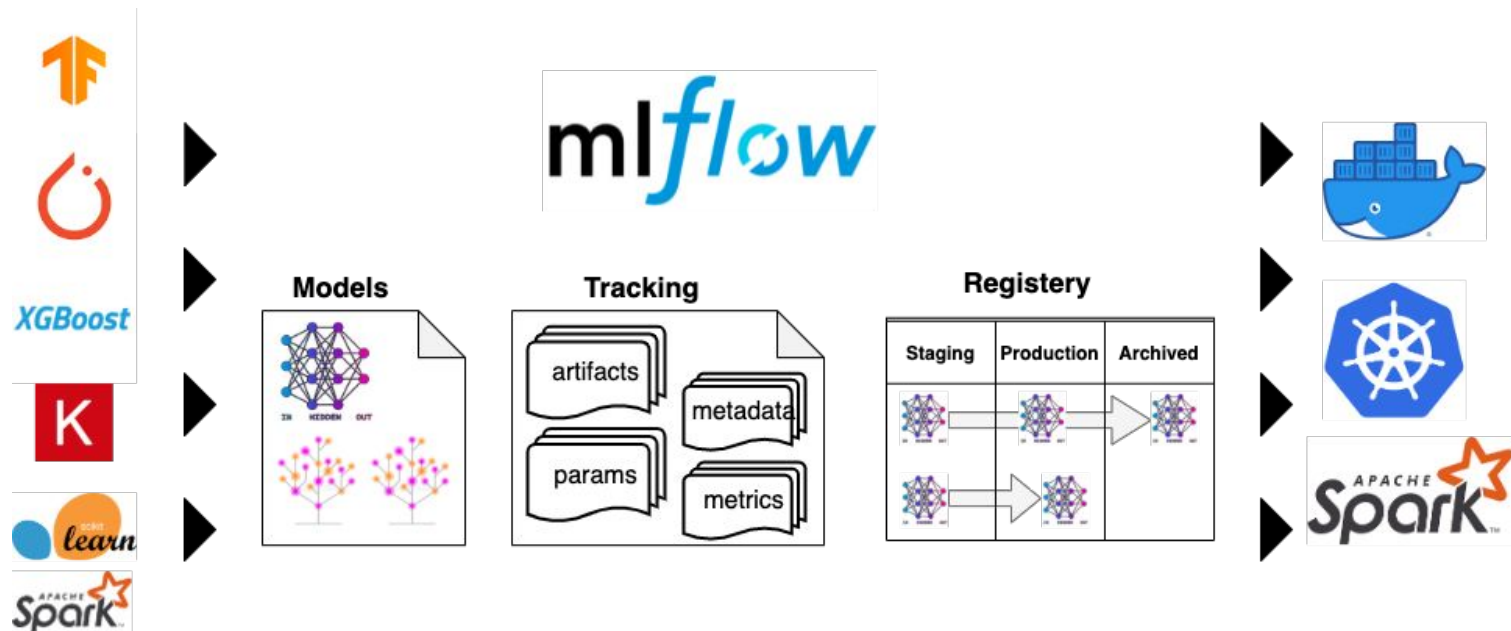
# MATRYCS: MLFlow tutorial

Gregor Cerar, PhD  
Consensus



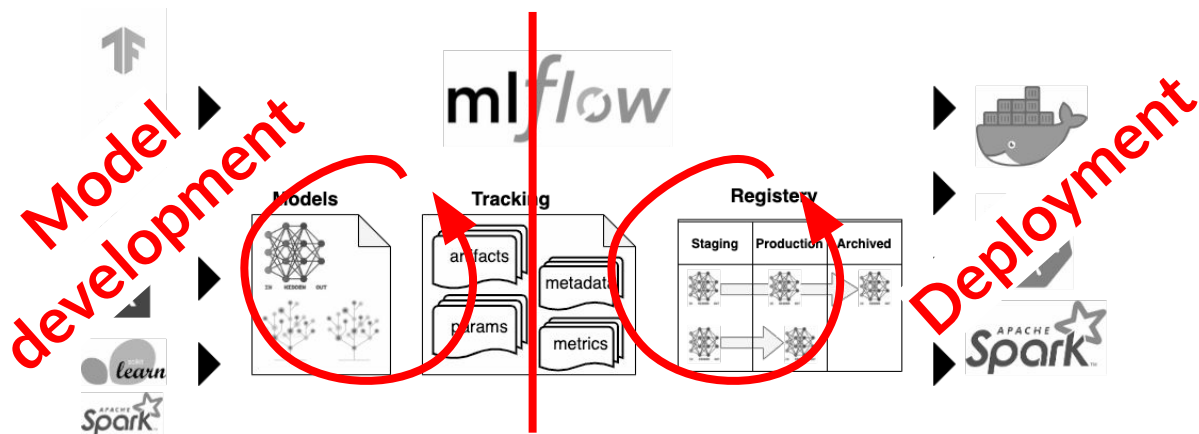
MATRYCS

## 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

The screenshot displays the MLFlow web interface. On the left, the 'Experiments' sidebar shows a list of experiments: 'Default', 'energy-estimation-1h', and 'dummy-regression'. The 'dummy-regression' experiment is selected. The main panel shows the details for 'dummy-regression', including a description and a table of attempts. The table has columns for Date, User, Source, Version, Parameters (alpha, l1\_ratio), and Metrics (mae, r2, rmse). The metrics are displayed as blue bars with numerical values.

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

## Registered Models > Airline\_Delay\_SparkML ▾

Created Time : 2019-10-10 15:20:29

Last Modified : 2019-10-14 12:17:04

### ▼ Description [🔗](#)

Predicts airline delays (in minutes) using the best Spark RF model from the AutoML Toolkit.

### ▼ Versions All Active(1)

Version	Registered at	Created by	Stage	Pending Requests
✔ <a href="#">Version 1</a>	2019-10-10 15:20:30	clemens@demo.com	Archived	—
✔ <a href="#">Version 2</a>	2019-10-10 21:47:29	clemens@demo.com	Archived	—
✔ <a href="#">Version 3</a>	2019-10-10 23:39:43	clemens@demo.com	Production	—
✔ <a href="#">Version 4</a>	2019-10-11 09:55:29	clemens@demo.com	None	—
✔ <a href="#">Version 5</a>	2019-10-11 12:44:44	matei@demo.com	Staging	1

# 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

```
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)
```

```
# 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

```
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)
```

dummy-regression > Run c2b0639603224b3fb74c0f17fc5b890b

Run c2b0639603224b3fb74c0f17fc5b890b

Date: 2022-04-08 09:00:14

Source: dummy-model.py

User: ubuntu

Duration: 1.1s

Status: FINISHED

Lifecycle Stage: active

Description [Edit](#)

Parameters

Metrics (3)

Name	Value
MAE <a href="#">🔗</a>	0
R2 <a href="#">🔗</a>	1
RMSE <a href="#">🔗</a>	0

Tags

Artifacts

- model
  - MLmodel
  - conda.yaml
  - model.pkl
  - requirements.txt
  - cdf.png

Full Path: mlflow-artifacts/2/c2b0639603224b3fb74c0f17fc5b890b/artifacts/model...  
Size: 394B

```
artifact_path: model
flavors:
  python_function:
    env: conda.yaml
    loader_module: mlflow.sklearn
    model_path: model.pkl
    python_version: 3.8.10
sklearn:
  pickled_model: model.pkl
  serialization_format: cloudpickle
  sklearn_version: 1.0.2
model_uuid: 9258da7ef2f64808ba29d3aa5a9370b2
run_id: c2b0639603224b3fb74c0f17fc5b890b
utc_time_created: '2022-04-08 09:00:14.410924'
```

## Example: Programmatically list all registered models

Create client instance

Access all registered models

Access latest versions of model

```
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

```
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](https://github.com/MATRYCS/ml_model_tracking_framework)