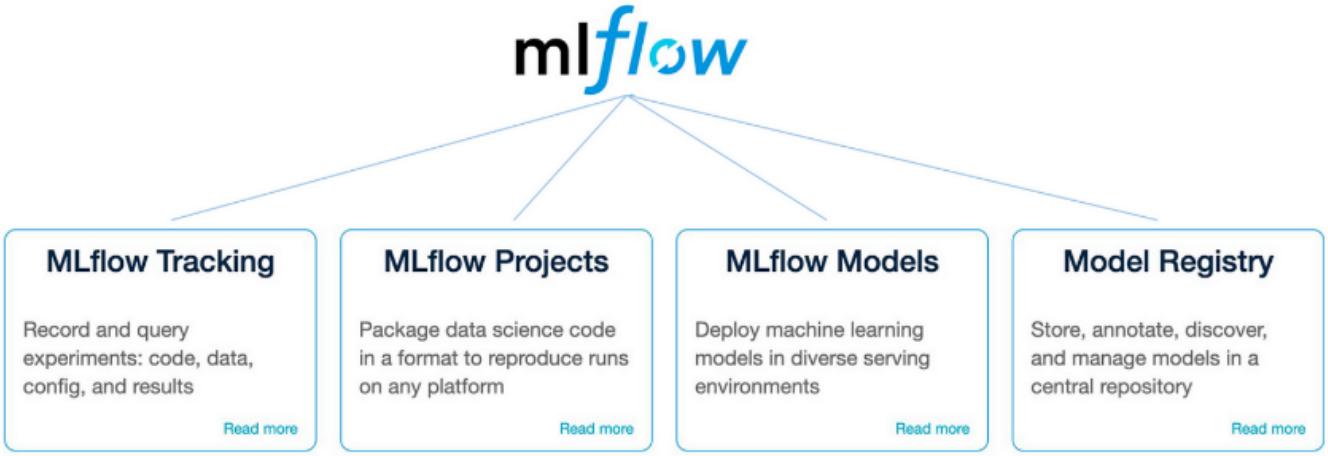
Credit 1: https://www.youtube.com/watch?v=6j_qhTJgB1w

Credit 2: https://github.com/mlflow/mlflow/blob/master/examples/sklearn_elasticnet_wine/train.ipynb

# What is MLflow?

MLflow is a framework that supports the machine learning lifecycle. This means that it has components to monitor your model during training and running, ability to store models, load the model in production code and create a pipeline.

ref: https://towardsdatascience.com/getting-started-with-mlflow-52eff8c09c61




# Install mlflow

```
!pip install mlflow --quiet --use-deprecated=legacy-resolver
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.5/18.5 MB 44.0 MB/s eta 0:
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 83.5/83.5 kB 9.9 MB/s eta 0:0
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 189.5/189.5 kB 18.7 MB/s eta 0:
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 226.0/226.0 kB 24.8 MB/s eta 0:
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 148.1/148.1 kB 13.3 MB/s eta 0:
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 80.2/80.2 kB 8.8 MB/s eta 0:0
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 143.1/143.1 kB 16.8 MB/s eta 0:
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.7/62.7 kB 8.3 MB/s eta 0:0
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 78.7/78.7 kB 10.2 MB/s eta 0:
  Building wheel for databricks-cli (setup.py) ... done
```

# Import libraries

```
# Importing all Libraries
import mlflow
import mlflow.sklearn
#mlflow.set_experiment('mlflow-demo')

import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score, accuracy_score, classification_report
import warnings
warnings.filterwarnings("ignore")
```

# Simple MLFLow Workflow for Scikit-Learn

1. Start an experiment using mlflow.start_run() which switches the context of your existing model code to enable mlflow tracking.
2. We log the run parameters with mlflow.log_param()
3. We log the model metrics (mean accuracy on the training set in this case) with mlflow.log_metric().
4. After model training and evaluation, I have logged the model using mlflow.sklearn.log_model().

```
# Load and split dataset
```

```
# Load and split dataset
X, Y = load_breast_cancer(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random
print("Training Data Shape: ", X_train.shape, y_train.shape)
print("Testing Data Shape: ", X_test.shape, y_test.shape)

local_registry = "sqlite:///mlruns.db"
mlflow.set_tracking_uri(local_registry)
experiment_id = mlflow.set_experiment('test_experiment')

def eval_metrics(actual, pred):
    accuracy = accuracy_score(actual, pred)
    return accuracy

def train_model(criterion, max_depth):

    # Starting the Experiement
    with mlflow.start_run():

        # Model building
        model = DecisionTreeClassifier(criterion=criterion, max_depth=max_depth
        model.fit(X_train, y_train)   # Model Training
        y_pred = model.predict(X_test)  # Model Prediction on Testing data
        (accuracy) = eval_metrics(y_test, y_pred)

        print('Decirion tree (criterion=%s, max_depth=%d):'%(criterion, max_dep
        print('Accuracy: {:.4f}'.format(accuracy))

        # Logging Parameters
        mlflow.log_param("criterion", criterion)
        mlflow.log_param("max_depth", max_depth)

        # Logging Metrics
        mlflow.log_metric("accuracy", accuracy_score(y_test, y_pred))

        # Model Logging
        mlflow.sklearn.log_model(model, 'model')

        return model
```

```
Training Data Shape:  (455, 30) (455,)
Testing Data Shape:  (114, 30) (114,)
2023/09/18 10:21:03 INFO mlflow.store.db.utils: Creating initial MLflow dat
2023/09/18 10:21:03 INFO mlflow.store.db.utils: Updating database tables
INFO  [alembic.runtime.migration] Context impl SQLiteImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
INFO  [alembic.runtime.migration] Running upgrade  -> 451aebb31d03, add met
INFO  [alembic.runtime.migration] Running upgrade 451aebb31d03 -> 90e64c465
INFO  [alembic.runtime.migration] Running upgrade 90e64c465722 -> 181f10493
INFO  [alembic.runtime.migration] Running upgrade 181f10493468 -> df50e92ff
INFO  [alembic.runtime.migration] Running upgrade df50e92ffc5e -> 7ac759974
INFO  [alembic.runtime.migration] Running upgrade 7ac759974ad8 -> 89d4b8295
INFO  [89d4b8295536_create_latest_metrics_table_py] Migration complete!
INFO  [alembic.runtime.migration] Running upgrade 89d4b8295536 -> 2b4d017a5
INFO  [2b4d017a5e9b_add_model_registry_tables_to_db_py] Adding registered_m
INFO  [2b4d017a5e9b_add_model_registry_tables_to_db_py] Migration complete!
INFO  [alembic.runtime.migration] Running upgrade 2b4d017a5e9b -> cfd24bdc0
INFO  [alembic.runtime.migration] Running upgrade cfd24bdc0731 -> 0a8213491
INFO  [alembic.runtime.migration] Running upgrade 0a8213491aaa -> 728d730b5
INFO  [alembic.runtime.migration] Running upgrade 728d730b5ebd -> 27a6a02d2
INFO  [alembic.runtime.migration] Running upgrade 27a6a02d2cf1 -> 84291f40a
INFO  [alembic.runtime.migration] Running upgrade 84291f40a231 -> a8c4a736b
INFO  [alembic.runtime.migration] Running upgrade a8c4a736bde6 -> 39d1c3be5
INFO  [alembic.runtime.migration] Running upgrade 39d1c3be5f05 -> c48cb773b
INFO  [alembic.runtime.migration] Running upgrade c48cb773bb87 -> bd07f7e96
INFO  [alembic.runtime.migration] Running upgrade bd07f7e963c5 -> 0c779009a
INFO  [alembic.runtime.migration] Running upgrade 0c779009ac13 -> cc1f77228
INFO  [alembic.runtime.migration] Running upgrade cc1f77228345 -> 97727af70
INFO  [alembic.runtime.migration] Running upgrade 97727af70f4d -> 3500859a5
INFO  [alembic.runtime.migration] Running upgrade 3500859a5d39 -> 7f2a7d5fa
INFO  [alembic.runtime.migration] Context impl SQLiteImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
2023/09/18 10:21:04 INFO mlflow.tracking.fluent: Experiment with name 'test
```

# Train 10 decision trees with different criterion and max depth

```
train_model('gini', 1)
```

```
Decirion tree (criterion=gini, max_depth=1):
Accuracy: 0.9035
```

▾ DecisionTreeClassifier

```
DecisionTreeClassifier(max_depth=1, random_state=0)
```

```
train_model('gini', 2)
```

```
Decirion tree (criterion=gini, max_depth=2):
Accuracy: 0.9649
```

```
DecisionTreeClassifier(max_depth=2, random_state=0)
```

```
train_model('gini', 3)
```

```
Decirion tree (criterion=gini, max_depth=3):
Accuracy: 0.9649
```

```
DecisionTreeClassifier(max_depth=3, random_state=0)
```

```
train_model('gini', 4)
```

```
Decirion tree (criterion=gini, max_depth=4):
Accuracy: 0.9561
```

```
DecisionTreeClassifier(max_depth=4, random_state=0)
```

```
train_model('gini', 5)
```

```
Decirion tree (criterion=gini, max_depth=5):
Accuracy: 0.9474
```

```
DecisionTreeClassifier(max_depth=5, random_state=0)
```

```
train_model('entropy', 1)
```

```
Decirion tree (criterion=entropy, max_depth=1):
Accuracy: 0.9035
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=1, random_state=0)
```

```
train_model('entropy', 2)
```

```
Decirion tree (criterion=entropy, max_depth=2):
Accuracy: 0.9211
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=2, random_state=0)
```

```
train_model('entropy', 3)
```

```
Decirion tree (criterion=entropy, max_depth=3):
Accuracy: 0.9474
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)
```

```
train_model('entropy', 4)
```

```
Decirion tree (criterion=entropy, max_depth=4):
Accuracy: 0.9386
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=4, random_state=0)
```

```
train_model('entropy', 5)
```

```
Decirion tree (criterion=entropy, max_depth=5):
Accuracy: 0.9211
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=5, random_state=0)
```

# MLflow Models

An MLflow Model is a standard format for packaging machine learning models that can be used in a variety of downstream tools

```
#Search best 5 runs
best_run_df = mlflow.search_runs(order_by=['metrics.accuracy DESC'], max_result
best_run_df
```

| | run_id | experiment_id | status | |
|---|---|---|---|---|
| **0** | 2a8ceb1089f843a1be2f2dfa89429b19 | 1 | FINISHED | /content/mlruns/1/2a |
| **1** | f181b6c5a9f84684b73cb573ef201a42 | 1 | FINISHED | /content/mlruns/1/f1 |
| **2** | ec9be1b6162348b0bcf91051a2283b27 | 1 | FINISHED | /content/mlruns/1/ec9 |
| **3** | 3bfc888eff7e498d838b0ded0f6e3eaa | 1 | FINISHED | /content/mlruns/1/3b |
| **4** | 93a928825e524110b94c52d74ab39aa8 | 1 | FINISHED | /content/mlruns/1/93a |

```
run_id = str(best_run_df.loc[0, 'run_id'])
print('run_id: ', run_id)

run_id = str(best_run_df.loc[0, 'run_id'])
model_uri = f"runs:/{run_id}/model"
print('model_uri: ', model_uri)
```

```
run_id:  2a8ceb1089f843a1be2f2dfa89429b19
model_uri:  runs:/2a8ceb1089f843a1be2f2dfa89429b19/model
```

```
# Load model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(model_uri=f"runs:/{run_id}/model")

# Predict on a Pandas DataFrame.
predicted = loaded_model.predict(pd.DataFrame(X_test))

print(classification_report(y_test, predicted, target_names=['Non-DD', 'DD'], c
```

```
                  precision    recall  f1-score   support

          Non-DD     0.9778    0.9362    0.9565        47
              DD     0.9565    0.9851    0.9706        67

        accuracy                         0.9649       114
       macro avg     0.9671    0.9606    0.9636       114
    weighted avg     0.9653    0.9649    0.9648       114
```

## Model Registry

The MLflow Model Registry component is a centralized model store, set of APIs, and UI, to collaboratively manage the full lifecycle of an MLflow Model. It provides model lineage, model versioning, stage transitions (for example from staging to production), and annotations.

ref: https://mlflow.org/docs/latest/model-registry.html

```
#Register best model
mlflow.register_model(model_uri=model_uri, name="breast_cancer")
```

```
Successfully registered model 'breast_cancer'.
2023/09/18 10:21:28 INFO mlflow.tracking._model_registry.client: Waiting up
Created version '1' of model 'breast_cancer'.
<ModelVersion: aliases=[], creation_timestamp=1695032488904,
current_stage='None', description=None,
last_updated_timestamp=1695032488904, name='breast_cancer',
run_id='2a8ceb1089f843a1be2f2dfa89429b19', run_link=None,
source='/content/mlruns/1/2a8ceb1089f843a1be2f2dfa89429b19/artifacts/model'
status='READY', status_message=None, tags={}, user_id=None, version=1>
```

## Test the model

```
model_name = "breast_cancer"
model_version = 1

# Load model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(model_uri=f"models:/{model_name}/{model

# Predict on a Pandas DataFrame.
predicted = loaded_model.predict(pd.DataFrame(X_test))
```

```
from sklearn.metrics import  classification_report
print(classification_report(y_test, predicted, target_names=['Non-DD', 'DD'], c
```

```
              precision    recall  f1-score   support

      Non-DD     0.9778    0.9362    0.9565        47
          DD     0.9565    0.9851    0.9706        67

    accuracy                         0.9649       114
   macro avg     0.9671    0.9606    0.9636       114
weighted avg     0.9653    0.9649    0.9648       114
```

# MLflow UI

After completing the model training and logging, we can track the model progress using MLFlow UI.

To enable the tracking, Navigate to the curselfrent project in Terminal and use the command below

Access this link: http://localhost:5000/

```
!pip install pyngrok --quiet
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 698.7/698.7 kB 11.9 MB/s eta 0:
  Preparing metadata (setup.py) ... done
  Building wheel for pyngrok (setup.py) ... done
```

```
from pyngrok import ngrok
ngrok.kill()

#Setting the authtoken (optional)
#Get your authtoken from https://dashboard.ngrok.com/auth
NGROK_AUTH_TOKEN = '' # Enter your authtoken
ngrok.set_auth_token(NGROK_AUTH_TOKEN)

# Open an HTTPs tunnel on port 5000 for http://localhost:5000
ngrok_tunnel = ngrok.connect(addr='5000', proto='http', bind_tls=True)
print("MLflow Tracking UI: ", ngrok_tunnel.public_url)
```

```
WARNI [pyngrok.process.ngrok] t=2023-09-18T10:21:33+0000 lvl=warn msg="ngro
MLflow Tracking UI:  https://49a1-34-122-254-241.ngrok-free.app
```

```
!mlflow ui --backend-store-uri sqlite:///mlruns.db
# Access this link: http://localhost:5000/
```

```
[2023-09-18 10:21:35 +0000] [776] [INFO] Starting gunicorn 21.2.0
[2023-09-18 10:21:35 +0000] [776] [INFO] Listening at: http://127.0.0.1:500
[2023-09-18 10:21:35 +0000] [776] [INFO] Using worker: sync
[2023-09-18 10:21:35 +0000] [778] [INFO] Booting worker with pid: 778
[2023-09-18 10:21:35 +0000] [779] [INFO] Booting worker with pid: 779
[2023-09-18 10:21:35 +0000] [780] [INFO] Booting worker with pid: 780
[2023-09-18 10:21:36 +0000] [781] [INFO] Booting worker with pid: 781
[2023-09-18 10:23:08 +0000] [776] [INFO] Handling signal: int

Aborted!
[2023-09-18 10:23:08 +0000] [781] [INFO] Worker exiting (pid: 781)
[2023-09-18 10:23:08 +0000] [779] [INFO] Worker exiting (pid: 779)
[2023-09-18 10:23:08 +0000] [778] [INFO] Worker exiting (pid: 778)
[2023-09-18 10:23:08 +0000] [780] [INFO] Worker exiting (pid: 780)
[2023-09-18 10:23:09 +0000] [776] [INFO] Shutting down: Master
```