ML FLOW REPORT

Name: Shiwani Dembla

CMS-ID: 023-18-0107

Subject: Data Science

Instructor: Mr. Syed Muzamil Hussain Shah

Contents

Introduction & Background

The goal of using Machine Learning is to optimize metrics (e.g., accuracy. Constantly experiment to improve it. The quality of models depends on input data and tuning parameters. We compare and combine many libraries, model and use a diverse deployment environment. Keeping all these functionalities easy to use and manage, MLflow was introduced.

MLflow is an open machine learning platform.

- It works with popular ML library & language
- Runs the same way anywhere (e.g., any cloud or locally)
- Designed to be useful for 1 or 1000+ person orgs
- Simple. Modular. Easy-to-use.
- Offers positive developer experience to get started

MLflow Design Philosophy

The working of MLflow is based on 2 different design philosophies which define its way of working.

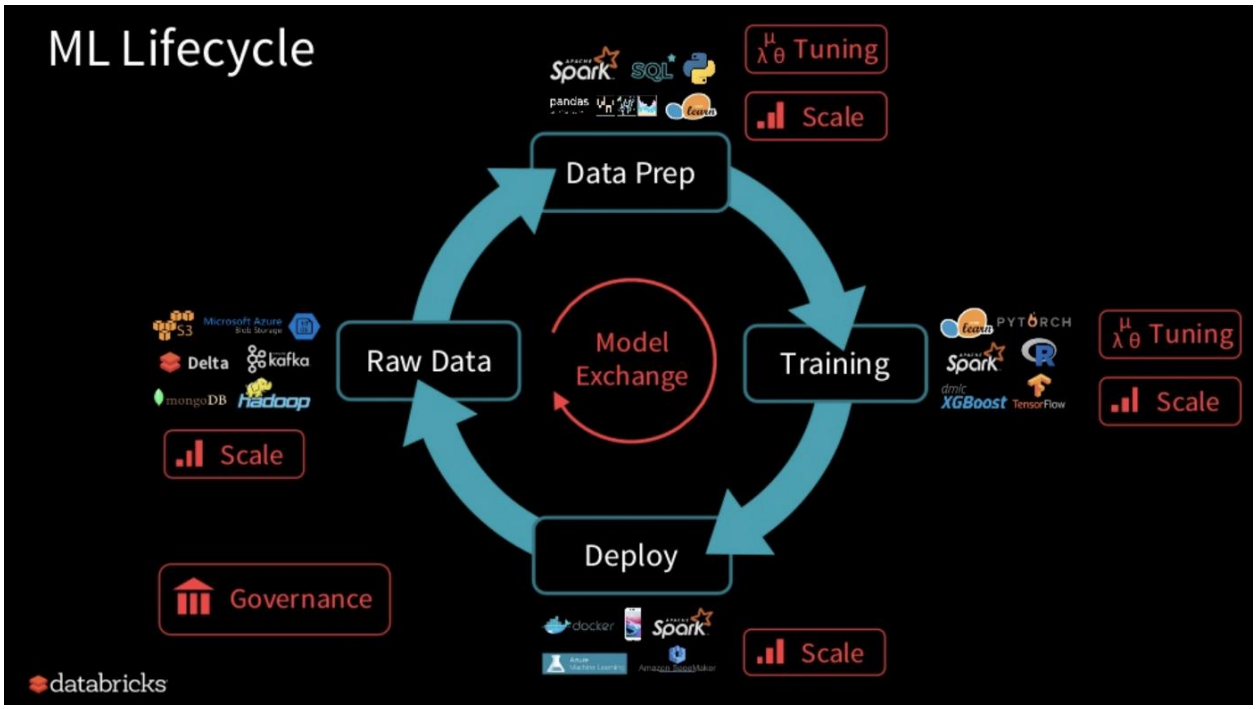
1. API First Design
    - Submit runs, log models, metrics, etc. from popular library & language
    - Abstract "model" lambda function that MLflow can then deploy in many places (Docker, Azure ML, Spark UDF)
    - Open interface allows easy integration from the community

    It is built around Programmatic APIs, REST APIs & CLI

2. Modular Design
    - Allow different components individually (e.g., use MLflow's project format but not its deployment tools
    - Not monolithic
    - But Distinctive and Selective

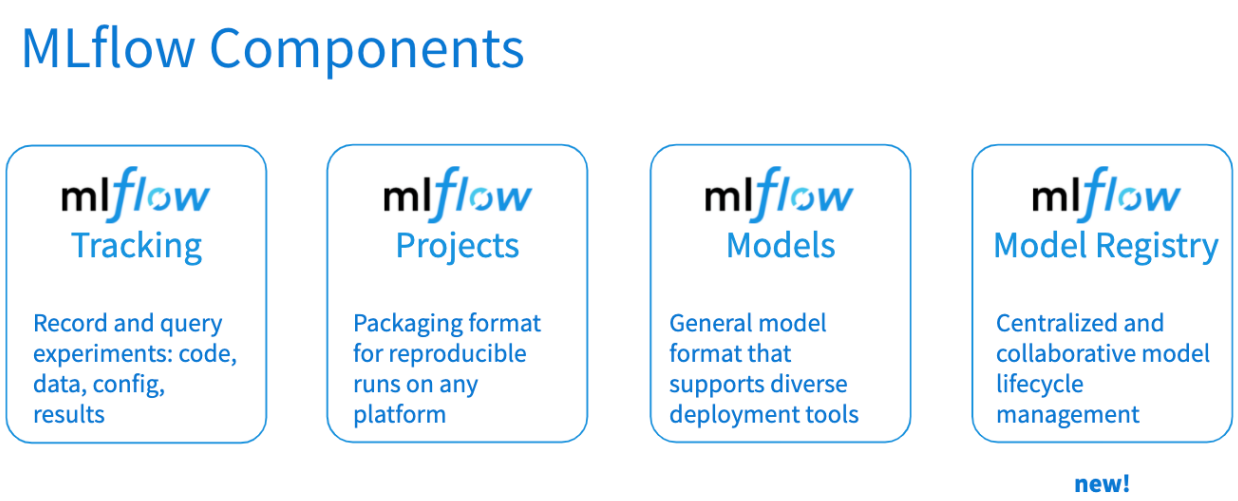
    Based on distinct components (Tracking/Projects/Models/Registry

MLflow Lifecycle



ML Lifecycle includes taking the raw data and cleaning it by handling missing and noisy values, transforming it into an organized, prepared data. Tuning and scaling is done on the Data. Different models are used for the training and deployment. Every stage of this life cycle becomes easy using MLflow lifecycle.

It includes mainly 4 components:
1.Tracking
2.Projects
3.Models
4.Registry



ML flow Projects

An MLflow Project is a format for packaging data science code in a reusable and reproducible way, based primarily on conventions. In addition, the Projects component includes an API and command-line tools for running projects, making it possible to chain together projects into workflows.

ML flow Models

An MLflow Model is a standard format for packaging machine learning models that can be used in a variety of downstream tools—for example, real-time serving through a REST API or batch inference on Apache Spark. The format defines a convention that lets you save a model in different “flavors” that can be understood by different downstream tools.

ML flow 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 (which MLflow experiment and run produced the model), model versioning, stage transitions (for example from staging to production), and annotations.

ML flow Tracking

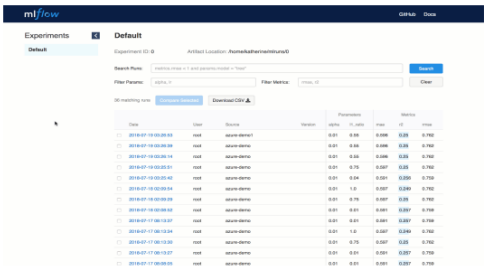
ML flow Tracking is an API and UI for logging parameters, code versions, metrics and output files when running your machine learning code to later visualize them. With a few simple lines of code, you can track parameters, metrics, and artifacts:

- Parameters: key-value inputs to your code
- Metrics: numeric values (can update over time)
- Tags and Notes: information about a run
- Artifacts: files, data, and models
- Source: what code ran?
- Version: what of the code?
- Run: an instance of code that runs by MLflow
- Experiment: {Run, … Run

Model Development *with* MLflow is *Simple!*

```
import mlflow
data   = load_text(file)
ngrams = extract_ngrams(data, N=n)
model  = train_model(ngrams,
             learning_rate=lr)
score  = compute_accuracy(model)
with mlflow.start_run():
  mlflow.log_param("data_file", file)
  mlflow.log_param("n", n)
  mlflow.log_param("learn_rate", lr)
  mlflow.log_metric("score", score)
  mlflow.sklearn.log_model(model)
```

```
$ mlflow ui
```



Track parameters, metrics, output files & code version

In [8]:
```
#install MLFlow

#!pip install mlflow
```

In [11]:
```
#Check whether MLFlow is installed or not

#!mlflow
```

In [10]:
```
#!python -m pip install --upgrade pip
```

In [13]:
```
#let's do practically
import mlflow
import mlflow.sklearn

mlflow.set_experiment('LearnML-Demo')
```

```
INFO: 'LearnML-Demo' does not exist. Creating a new experiment
```

In [28]:
```
import pandas as pd
import numpy as np
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import ElasticNet
import mlflow
import mlflow.sklearn
import sys
import os
```

In [29]:
```
def eval_metrics(actual, pred):
        rmse = np.sqrt(mean_squared_error(actual, pred))
        mae = mean_absolute_error(actual, pred)
        r2 = r2_score(actual, pred)
        return rmse, mae, r2
```

In [31]:
```
data = pd.read_csv("winequality-red.csv",delimiter=";")
```

In [32]:
```
data.head()
```

Out[32]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

In [35]:
```
# Read the wine-quality csv file


def train_model(alpha,l1_ratio):
    train, test = train_test_split(data)

    # The predicted column is "quality" which is a scalar from [3, 9]
    train_x = train.drop(["quality"], axis=1)
    test_x = test.drop(["quality"], axis=1)
    train_y = train[["quality"]]
    test_y = test[["quality"]]

    with mlflow.start_run():
        lr = ElasticNet(alpha=alpha, l1_ratio=l1_ratio, random_state=42)
        lr.fit(train_x, train_y)

        predicted_qualities = lr.predict(test_x)

        (rmse, mae, r2) = eval_metrics(test_y, predicted_qualities)
        print("Elasticnet model (alpha=%f, l1_ratio=%f):" % (alpha, l1_ratio))
        print("  RMSE: %s" % rmse)
        print("  MAE: %s" % mae)
        print("  R2: %s" % r2)
```

```
    # The predicted column is "quality" which is a scalar from [3, 9]
    train_x = train.drop(["quality"], axis=1)
    test_x = test.drop(["quality"], axis=1)
    train_y = train[["quality"]]
    test_y = test[["quality"]]

    with mlflow.start_run():
        lr = ElasticNet(alpha=alpha, l1_ratio=l1_ratio, random_state=42)
        lr.fit(train_x, train_y)

        predicted_qualities = lr.predict(test_x)

        (rmse, mae, r2) = eval_metrics(test_y, predicted_qualities)
        print("Elasticnet model (alpha=%f, l1_ratio=%f):" % (alpha, l1_ratio))
        print("  RMSE: %s" % rmse)
        print("  MAE: %s" % mae)
        print("  R2: %s" % r2)

        mlflow.log_param("alpha", alpha)
        mlflow.log_param("l1_ratio", l1_ratio)
        mlflow.log_metric("rmse", rmse)
        mlflow.log_metric("r2", r2)
        mlflow.log_metric("mae", mae)

        mlflow.sklearn.log_model(lr, "model")
```

In [36]:
```
train_model(0.5,0.5)
```

In [37]:
```
train_model(0.2,0.2)
```

In [38]:
```
train_model(0.1,0.1)
```

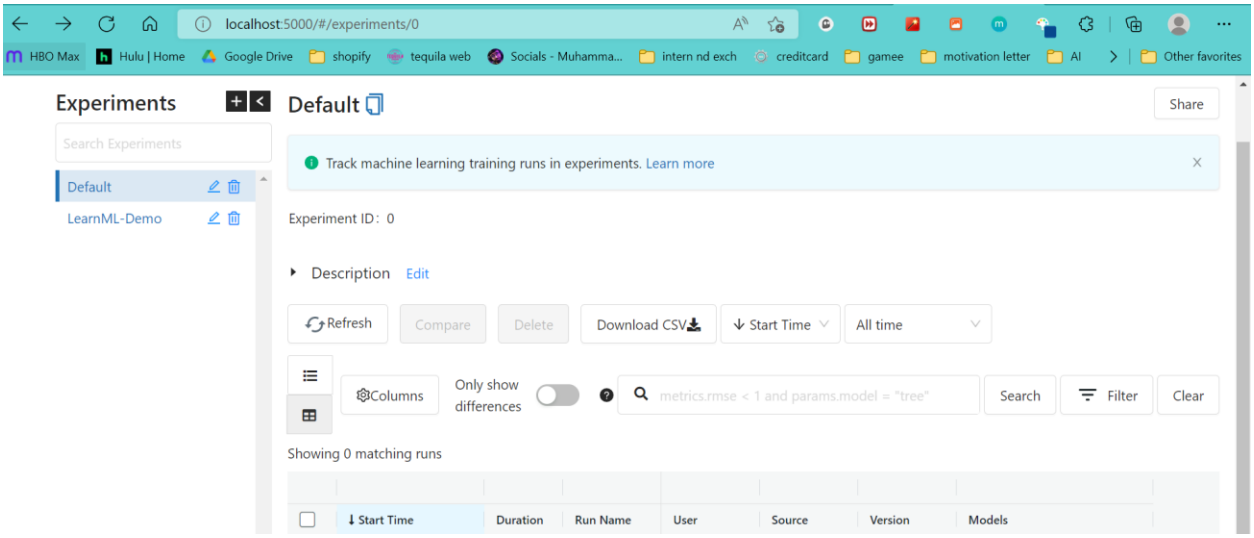
In [39]:
```
train_model(0.4,0.1)
```

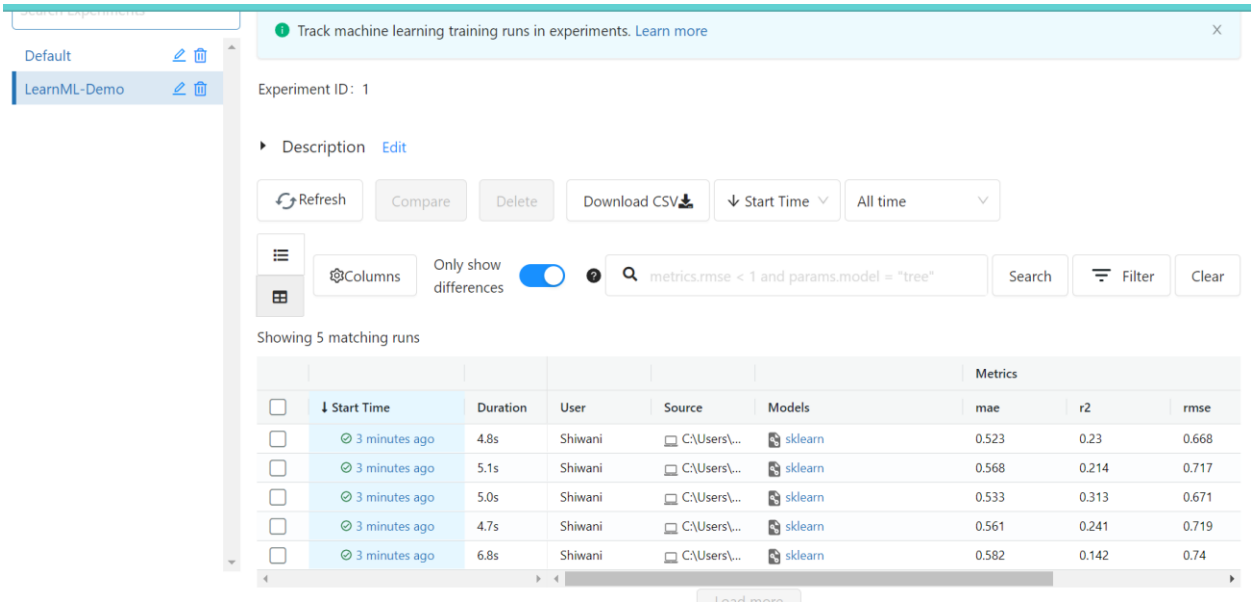
In [40]:
```
train_model(0.1,0.4)
```

When you run
!mlflow ui
Then you can go to your browser and go to

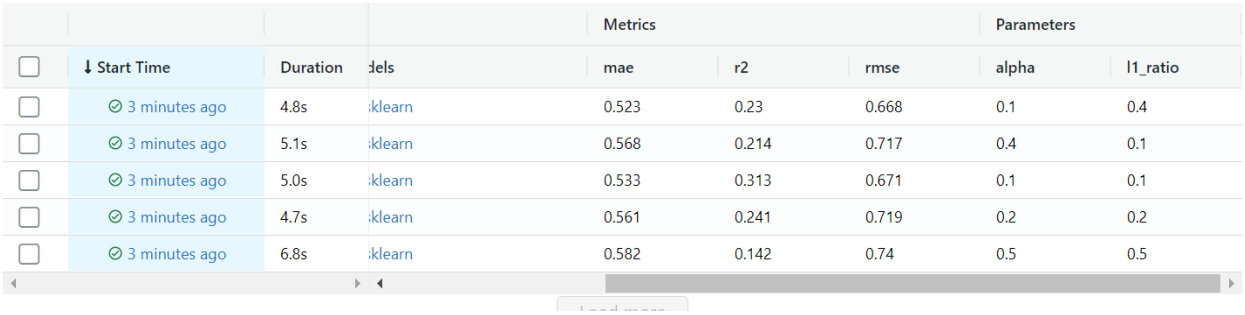
http://localhost:5000/

It gives you the UI of the tracking of the project showing the experimentation of ML models





It becomes easy to get notice of all the parameters, time taken and accuracy of a certain model.



				Metrics			Parameters	
	↓ Start Time	Duration	dels	mae	r2	rmse	alpha	l1_ratio
	3 minutes ago	4.8s	sklearn	0.523	0.23	0.668	0.1	0.4
	3 minutes ago	5.1s	sklearn	0.568	0.214	0.717	0.4	0.1
	3 minutes ago	5.0s	sklearn	0.533	0.313	0.671	0.1	0.1
	3 minutes ago	4.7s	sklearn	0.561	0.241	0.719	0.2	0.2
	3 minutes ago	6.8s	sklearn	0.582	0.142	0.74	0.5	0.5