**End-to-End Development of DNNs based Application Development**

* Complete ML Application Development Lifecycle
  + Model Development
  + Model Packaging
  + Model Deployment
  + Model Tracking
  + Model Monitoring / Observability
* Model Packaging – Docker Containerization
* Model Deployment – Kubernetes Deployment
* Model Tracking / Monitoring – MLOps

The business use-case is on “NLP” based RNNs & LSTM – Complete Workflow – Sentiment Analysis, Image Classification.

Common Machine Learning – Lifecycle Challenges during the Development:

* It’s difficult to keep track of ML Experiments
* It’s difficult to re-produce the code
* There’s no standard way to package and deploy models
* There’s no central store to manage models (Model versions and stage transitions – for example, moving from version 2 back to version 1 due to portability issues)

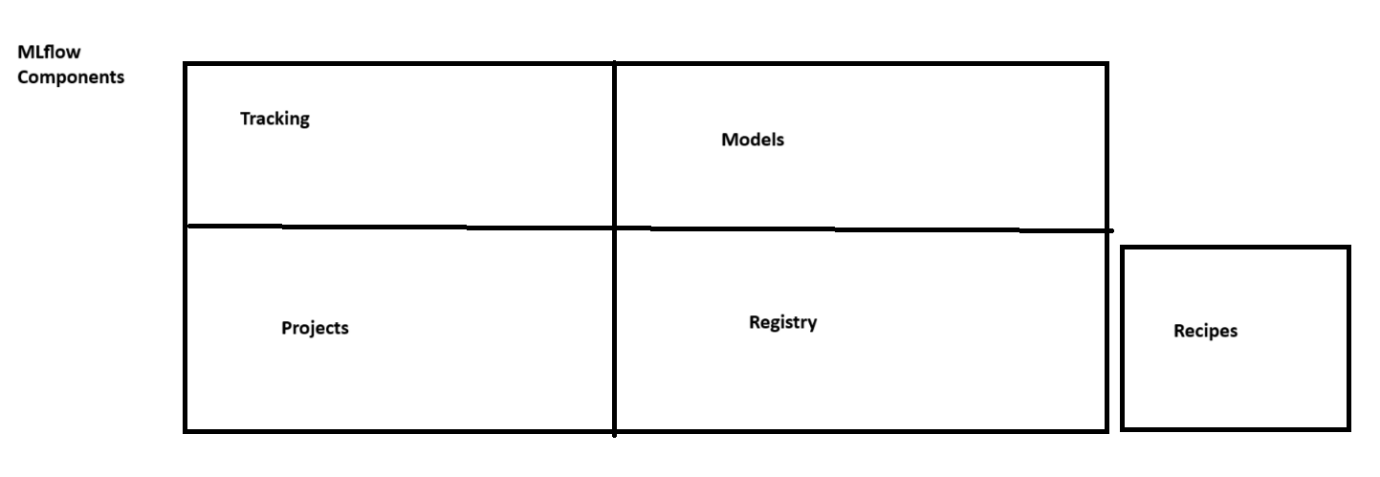
MLFlow

* It’s an open source framework designed to manage the end-to-end machine learning application development lifecycle.
* It provides several tools and utilities to help data scientists and ML engineers organize their work.
* It’s introduced by Databricks, which came up with Spark to solve Big Data Analytics enterprise challenges.
* It’s language agnostic – Supports Python, R, Java or REST APIs
* The primary goal of this open source framework is to provide
  + Tracking Experiments
  + Model Packaging
  + Model Deployment
  + Model Monitoring, Observability, Logging, Metrics …

Best Model Management Tools

* **MLFlow**
* DVC (Data Version Control)
* **Kubeflow**
* **TFX (Tensor Flow Extended)**
* Comet ML
* Neptune AI

MLFlow Concepts



**Components & Concepts in Detail**

* MLFlow Tracking
  + It’s an API as part of MLFlow and UI for logging parameters, code versions, metrics, datasets and artifacts when you run a ML workflow code and later it can be used to visualize the results.
  + Either those artifacts can be maintained in a local storage / centralized database such as PostgreSQL.
  + It keeps track of all experiments, related artifacts, which can then be visualized later and compared for better understanding
  + It also has the ability to store datasets used in Training / Evaluation and Testing purposes, and can be large-scale as long as the storage associated with MLFlow is big-data scale.
* MLFlow Projects
  + They’re a standard format for packaging re-usable data science code. Each project is simply a directory with code / or a GIT repository, and users a descriptor file and follows a simple convention to specific Python / Conda environments, with versions, libraries and dependency packages on how to run the code in a given environment. This guarantees to be able to replicate what data science team had in their environment to successfully producing the result, and replicate the same in other environments, without having to keeping track of all artifacts.
* MLFlow Models
  + It offers a conventional / standard for packaging ML Models in multiple flavors, and a variety of tools provided to help you to be able to package and deploy your ML models. Each model is served as a directory of arbitrary files and descriptors which describe how models are built, versioned, packaged, libraries, dependencies and so on. This simplifies how you do packaging irrespective of whether you use Tensor Flow, MXNET or any types of ML libraries.
* MLFlow Registry
  + It’s a model store, using which experimented models can be labelled, versioned and maintained in a repository. So later it can be published for use, and can be even rolled back to previous versions.
  + It’s a centralized model repository, APIs and UI, so it can be collaboratively manage the full lifecycle of an MLFlow project / Model.

Scalability and Big Data

* It’s designed to scale on Large Datasets, large output files and large number of experimented artifacts.
* MLFlow specifically supports four dimensions for Scalability and Big Data Support
  + MLFlow can run on a Local Server / Distributed Clustered Environment such as K8S, Cloud Service Providers (such as Azure ML, AWS Sagemaker)
  + Even if you run your experiments in a distributed cluster (for example, multiple GPUs), MFLow can be able to track all metrics, logs, tracing information.
  + MLFlow supports launching multiple runs in parallel with different languages / different parameters, and all are tracked in a centralized store.
  + MLFlow projects can be associated with appropriate storage services such as AWS S3 Storage, Azure Blob Storage or any Big Data Storage Services
  + MLFlow provides a centralized store to keep track hundrends of models, each model experiments, versions, runs, artifacts and stage transitions.