Data Science in production

Lecture 3: Versioning in a Data Science project

Alaa BAKHTI

Motivation

Model improvement: in a new data science iteration, we want to use the existing data to improve the production model (same data used to train the model)

- Make sure that the new model performs at least better than the production model on the same data
  - We need to track the model performance (Metrics versioning)
  - We need to track the data used to train and evaluate the previous model so that we can compare the 2 models (Data versioning)
- Treat the production of the model as a repeatable experiment (code, data, hyper-parameters, etc)
- Link the model to the training data, for example to investigate problems in productions and to understand the model defects

Motivation

Model retraining: new data is available, possibility of concept drift
The goal is not to improve the model performance on the past data but to make it up to date with the production data

- Update the initial training and validation sets with the new data
- Link the model to the training data, for example to investigate problems in productions and to understand the model defects
  - Why the model is no longer performing well after retraining? data quality problems?
- Be able to audit the model, for example when it takes decisions based on sensitive data

Data versioning

Data versioning

2 situations may occur depending on how the dataset is fed

- Data is immutable and only new data is appended
  - For example, sensors data (time series)
  - **How to version?** Keep a reference of the data used for training and validation (time ranges, filtering queries, IDs, ...)
  - **Problem**: complex to put in place, data model should not change over time so that the queries will remain valid, ...
- New data is appended while past data is updated
  - For example in time series data, we may update the values of past records
  - No guarantee that past data that was used to train the model will not change in the future
  - **How to version?**
    - Store the changes in a file to be able to reconstruct the dataset
    - take snapshots of the datasets before using them (training) and store it in a safe place
  - **Problem**: complex or high volume 😕

Some tools for data versioning

- Git
  - Not practical for large data snapshots: consider **Git LFS** or another tool. In Git-LFS, the data is stored in a seperate server.
  - If the dataset is binary (parquet file for example), git will not store only the difference between old and new dataset but will store a snapshot for the new dataset
  - Not optimized for network transfer of large datasets
- MLflow (very light)
  - Store modest datasets as experimental artifacts
  - Store the reference of the dataset (path in the datalake for example) in the properties of the experiment (tag)
- Databricks with Deltalake's *time travel* facility
- DVC
  - git-like semantics (checkout, commit, push, etc)

Inference data

- Why?
  - Reproduce the model prediction
  - Investigating model predictions
  - Analysing model errors
  - Auditing purposes
- What?
  - The data used for inference: data versioning (eg: min and max timestamp for time series data)
  - The model (id, name, version)
  - Model hyper-parameters: threshold, observation window, etc
  - Inference metadata: prediction time, etc
  - Model computed values: rule based model example
- How?
  - In a SQL database

Experiments tracking

Experiments tracking

Why

- Many experiments during the exploration and model building phase
- Reproduce the model (model and the achieved performances)

Tools:

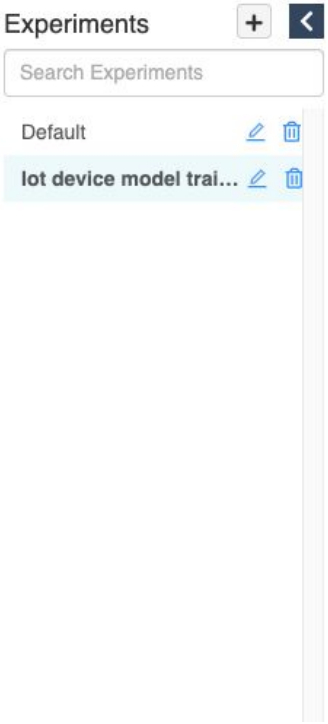
- MLflow
- CometML
- Aim
- DVC (V2.0), ...

What:

- Model, hyper-parameters, features, performances, ... + code & data

MLflow

- Open source platform for managing the end-to-end machine learning models lifecycle: the conception to its final stage of deployment
- Can be used to track models from different ML frameworks (scikit-learn, tensorflow, XGBoost, etc)
- Composed of
  - **Tracking**: to record and query experiments: code, configs, results, metrics, ...
  - **Models**: to manage and deploy models from a variety of ML libraries to a variety of model serving and inference platforms
  - **Projects**: to package ML code in a reusable, reproducible form to share with other data scientists or transfer to production.
  - **Model Registry**: to centralize a model store for managing models' full lifecycle stage transitions: from staging to production, with capabilities for versioning and annotating.
  - **Model Serving**: to host MLflow Models as REST endpoints.



Iot device model training

Experiment ID : 1

Artifact Location : /tmp/mlruns/1

▾ Notes

None

Search Runs: metrics.rmse < 1 and params.model = "tree" and tags.mlflow.source.type = "LOCAL"

State: Acti

Showing 4 matching runs | Compare | Delete | Download CSV

						Parameters >			Metrics >		
	Start Time	Run Name	User	Source	Version	bootstrap	ccp_alpha	criterion	test_mae	test_mse	te
	2021-07-28 16:16:35	-	alaa.bakhti	ipykernel_	-	True	0.0	mse	1.442	5.287	2.
	2021-07-28 16:16:32	-	alaa.bakhti	ipykernel_	-	True	0.0	mse	1.405	4.812	2.
	2021-07-28 16:16:28	-	alaa.bakhti	ipykernel_	-	True	0.0	mse	1.335	4.512	2.
	2021-07-28 16:16:25	-	alaa.bakhti	ipykernel_	-	True	0.0	mse	1.353	4.947	2.

MLflow Tracking

What can be tracked and how?

- **Parameters** with *mlflow.log_param*: Key-value input (model hyper-parameters, features, ...)
- **Metrics** with *mlflow.log_metric*: model performances (precision, recall, ...)
- **Artifacts** with *mlflow.log_artifact* or *mlflow.<framework>.log_model* for fitted model : files, including data and models
- Source, Version, Tags, ...
- For more informations: tracked concepts, logging functions
- It is possible to log all these informations with MLflow automatic logging
  - *mlflow.autolog()*
  - *mlflow.<framework>.autolog()* > *mlflow.sklearn.autolog()*

How to visualize the different experiments?

- With MLflow tracking UI

MLflow UI

- Visualize the list of experiment runs with the logged metadata (metrics, model parameters, etc)
- Search runs using the search filter API (simplified version of the SQL WHERE clause)
  - *metrics.accuracy > 0.92*
  - *metrics.rmse < 0.9 AND params.alpha = '1.0'*
- Compare models between themselves

Ressources

- Versioning in Machine learning delivery - OCTO Technology (french)
- MLflow guide - Databricks
- DVC getting started - DVC

Practical work