Model Development

Robert Clements

MSDS Program

University of San Francisco

We’ll Start with Model Development and Talk About the Data Later

- Goal: to learn how to approach the development of a machine learning model like a scientist.

- How: in the lab we will grab some freely available ML datasets and use a popular tool called MLFlow to conduct “experiments”.

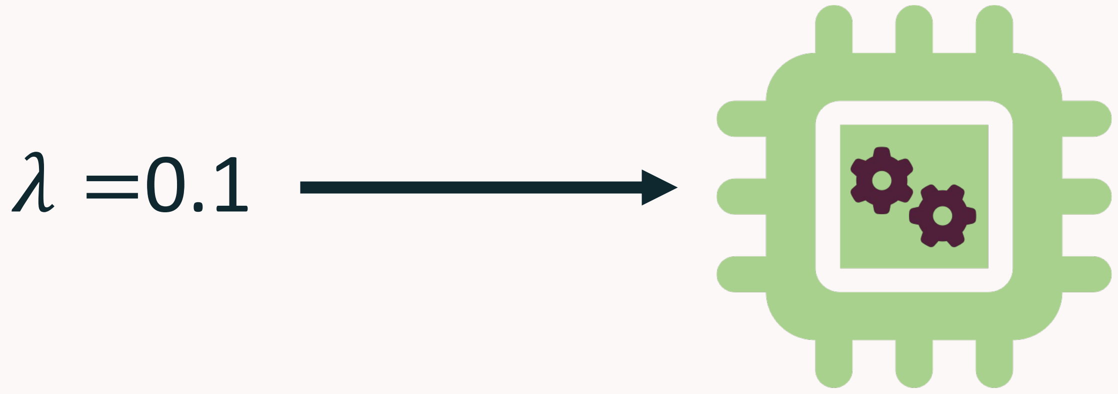
- Note: we are ***not*** going to be all that concerned with the actual model that we build. That is not the point here.

Experiment Tracking

How do YOU keep track of your models during model development?

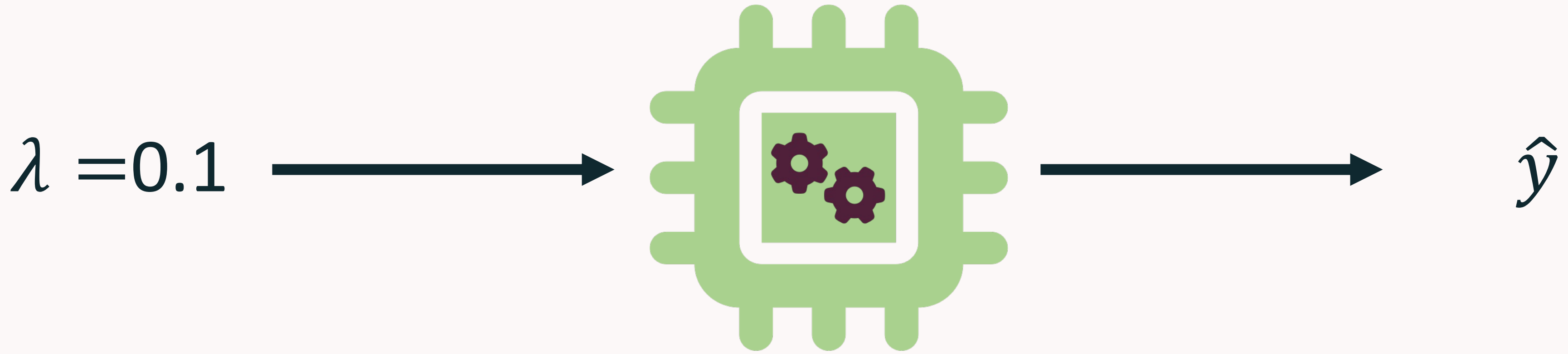
Doing Experiments

- Not talking about Experimentation (A/B testing, Multi-armed Bandit)
- Main idea:
  - Each model you train is a **run** of an **experiment**, with a specific treatment (parameter values, dataset, etc.).



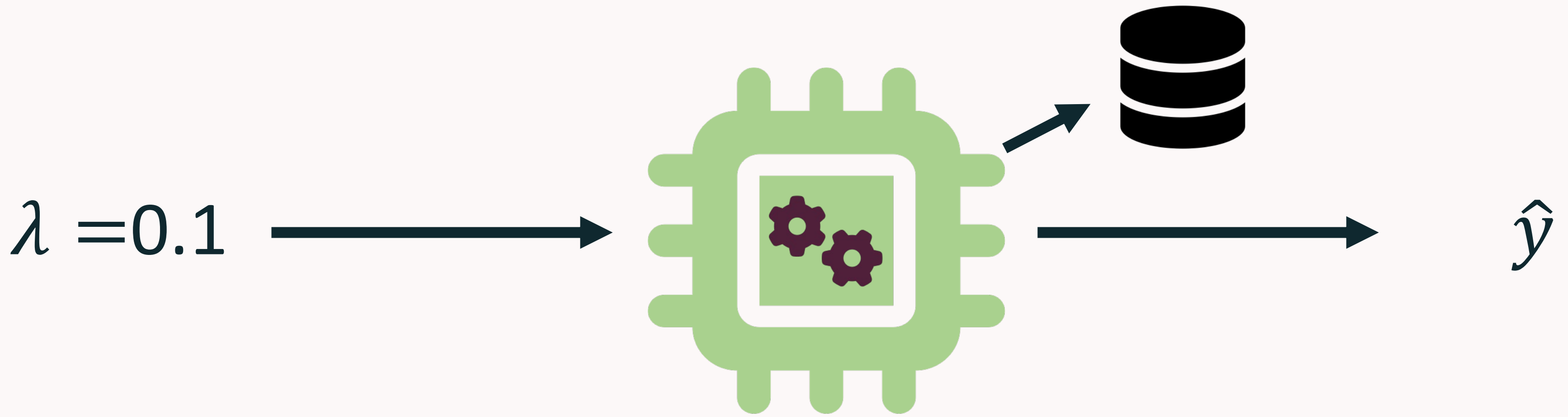
Doing Experiments

- Not talking about Experimentation (A/B testing, Multi-armed Bandit)
- Main idea:
  - Each model you train is a **run** of an **experiment**, with a specific treatment (parameter values, dataset, etc.).
  - Each run has a different **outcome** or **response** (model's parameter estimates, predictions) which is then used for assessing performance.



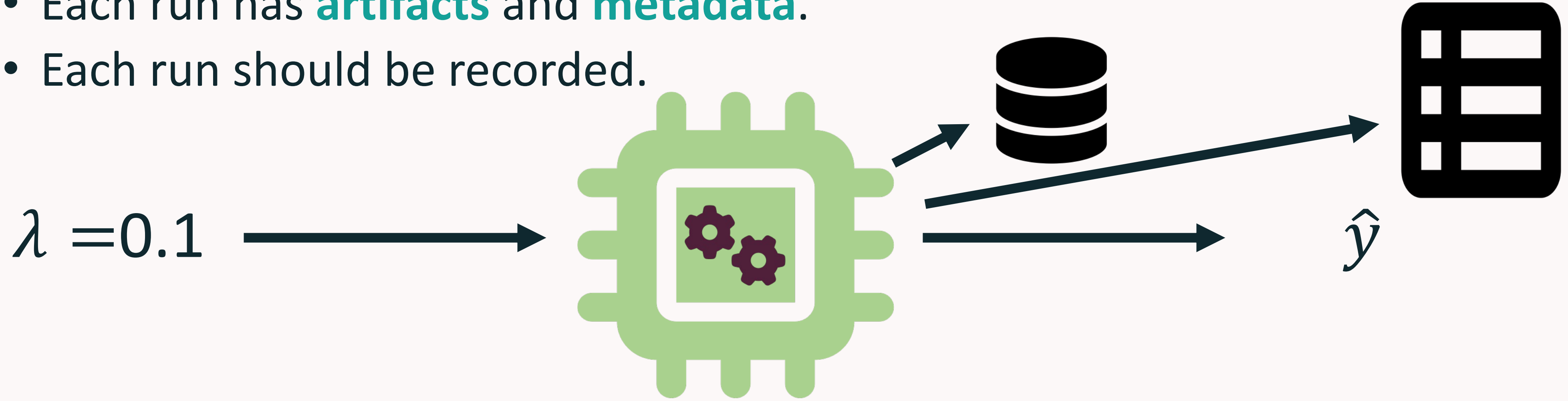
Doing Experiments

- Not talking about Experimentation (A/B testing, Multi-armed Bandit)
- Main idea:
  - Each model you train is a **run** of an **experiment**, with a specific treatment (parameter values, dataset, etc.).
  - Each run has a different **outcome** or **response** (model's parameter estimates, predictions) which is then used for assessing performance.
  - Each run has **artifacts** and **metadata**.



Doing Experiments

- Not talking about Experimentation (A/B testing, Multi-armed Bandit)
- Main idea:
  - Each model you train is a **run** of an **experiment**, with a specific treatment (parameter values, dataset, etc.).
  - Each run has a different **outcome** or **response** (model's parameter estimates, predictions) which is then used for assessing performance.
  - Each run has **artifacts** and **metadata**.
  - Each run should be recorded.



What Should We Track in our Experiments?

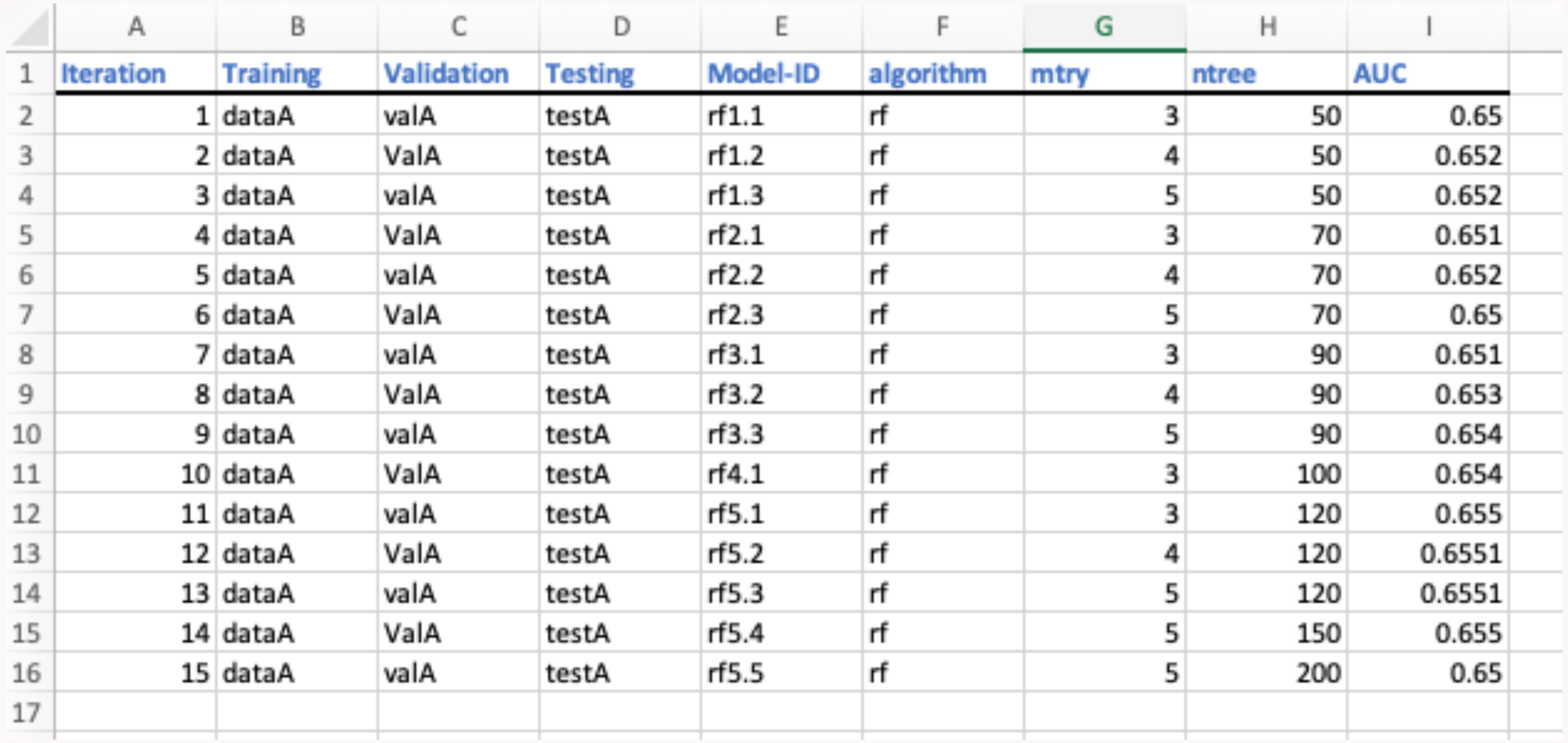
- Training, validation, and testing dataset versions
- Feature subsets
- Model hyperparameters
- Code changes
- Model evaluation metrics
- Model parameters/weights
- Score distributions
- Environment config files
- Feature importance
- Prediction explanations
- If doing deep learning: resource usage, gradient norms, predictions after each epoch (CV)

Examples

- You randomly split your data into a training and testing set using a random seed:
    - You should log the seed and version your training, validation and testing sets.
- You do hyperparameter tuning to find the optimal model:
    - You should log all results from each model (metric, performance charts, model weights, hyperparameter values) for each combination of hyperparameter values.
- You decide to try a different algorithm:
    - You should log all results of this model (metric, performance charts, model weights, hyperparameter values) to compare with your other models.

Why do we need a new tool to do this for us?

- It is difficult, and prone to error, to track results by porting them over to a spreadsheet.



Iteration	Training	Validation	Testing	Model-ID	algorithm	mtry	ntree	AUC
1	dataA	valA	testA	rf1.1	rf	3	50	0.65
2	dataA	ValA	testA	rf1.2	rf	4	50	0.652
3	dataA	valA	testA	rf1.3	rf	5	50	0.652
4	dataA	ValA	testA	rf2.1	rf	3	70	0.651
5	dataA	valA	testA	rf2.2	rf	4	70	0.652
6	dataA	ValA	testA	rf2.3	rf	5	70	0.65
7	dataA	valA	testA	rf3.1	rf	3	90	0.651
8	dataA	ValA	testA	rf3.2	rf	4	90	0.653
9	dataA	valA	testA	rf3.3	rf	5	90	0.654
10	dataA	ValA	testA	rf4.1	rf	3	100	0.654
11	dataA	valA	testA	rf5.1	rf	3	120	0.655
12	dataA	ValA	testA	rf5.2	rf	4	120	0.6551
13	dataA	valA	testA	rf5.3	rf	5	120	0.6551
14	dataA	ValA	testA	rf5.4	rf	5	150	0.655
15	dataA	valA	testA	rf5.5	rf	5	200	0.65

Why do we need a new tool to do this for us?

- It is difficult, and prone to error, to track results by porting them over to a spreadsheet.
- You do not commit all code changes – that is not efficient – and so you are bound to forget.
- Github does not help you compare models, or objects, even though you *can* version the results of your experiments in an automated way using git hooks.
- Reproducing results is hard to do without help. You absolutely will struggle to reproduce a good result you had without proper tracking. **You will forget**.

Experiment Tracking Tools

- MLFlow and Weights & Biases arguably the most popular

- We'll use MLFlow
  - Free to use
  - Large user community
  - Integrated with several other ML platforms such as Databricks and Dagshub

- Neptune and Comet are paid tools with more features
  - Neptune has a comparison of tools here: https://neptune.ai/blog/best-ml-experiment-tracking-tools

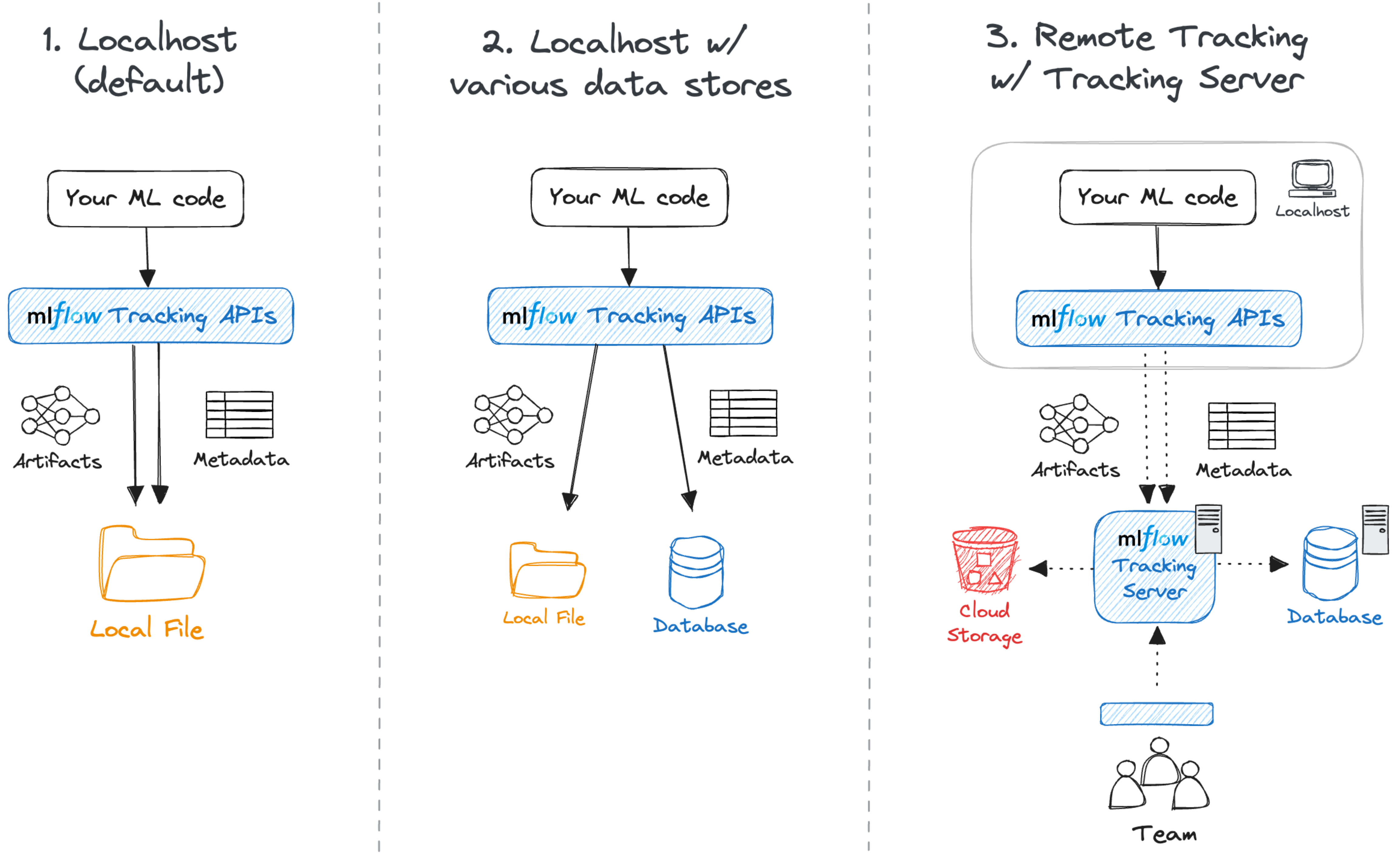
MLFlow

- MLFlow is a tool that promises to do four things:
  - Track experiments
  - Package projects
  - Store artifacts
  - Deploy models

- MLFlow is a python library:
  - Needs a backend store
  - Can run locally or on a server

MLFlow Architecture

- Where should everything be recorded?
- Options:
  - Local file system on your laptop.
  - Local file system (artifacts) + local sqlite (entities).
  - Remote tracking server (in GCP for instance):
    - Storage for artifacts and entities can be local or remote
    - If remote, maybe S3 for artifacts and postgres for entities
    - Good explanation here


1. Localhost (default)
Your ML code
mlflow Tracking APIs
Artifacts
Metadata
Local File
2. Localhost w/ various data stores
Your ML code
mlflow Tracking APIs
Artifacts
Metadata
Local File
Database
3. Remote Tracking w/ Tracking Server
Your ML code
Localhost
mlflow Tracking APIs
Artifacts
Metadata
Cloud Storage
mlflow Tracking Server
Database
Team

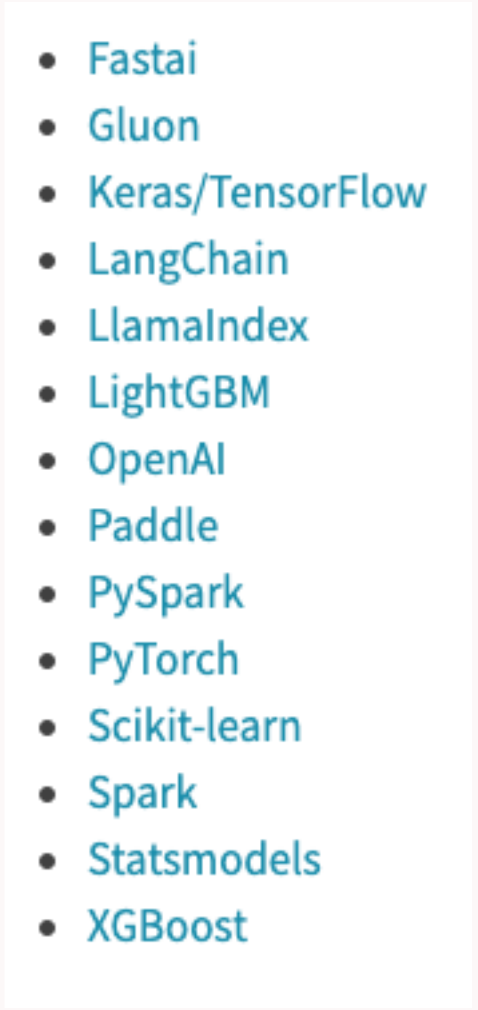
Experiment Tracking in MLFlow

```
Import mlflow

with mlflow.start_run():
     …code…

     mlflow.set_tag(“Tag_name”, “Tag value”)
     mlflow.log_params(params)
     mlflow.log_metric(“metric_name”, metric_value)
     mlflow.end_run()

OR

     mlflow.autolog() # can customize what gets logged
     mlflow.sklearn.autolog() # specific to sklearn
```

Experiment Tracking in MLFlow

```
Import mlflow

with mlflow.start_run():
     …code…

     mlflow.set_tag(“Tag_name”, “Tag value”)
     mlflow.log_params(params)
     mlflow.log_metric(“metric_name”, metric_value)
     mlflow.end_run()

OR

     mlflow.autolog() # can customize what gets logged
     mlflow.sklearn.autolog() # specific to sklearn
```

Autologging exists for all of these



- Fastai
- Gluon
- Keras/TensorFlow
- LangChain
- LlamaIndex
- LightGBM
- OpenAI
- Paddle
- PySpark
- PyTorch
- Scikit-learn
- Spark
- Statsmodels
- XGBoost

MLFlow Experiment Tracking Demo

Artifact Tracking and Model Registry

What are artifacts?

- An artifact is any output file that you'd like to store from the runs of your experiments:

Model File

Data

Plots/images

What are artifacts?

- An artifact is any output file that you'd like to store from the runs of your experiments:

Plots can be stored as image files.

You will create lots of images during model dev:

- EDA visualizations
- Model performance charts (ROC plots, PR curves, etc.)

Not everything needs to be stored, but these images can come in handy for reporting purposes.

Plots/images

Model File

Data

What are artifacts?

- An artifact is any output file that you'd like to store from the runs of your experiments:

Datasets can be stored in a compressed format such as parquet, particularly specific training, validation and test sets used.

The data sets can come in handy later, especially if something goes wrong with your deployed model.

Model File

Data

Plots/images

What are artifacts?

- An artifact is any output file that you'd like to store from the runs of your experiments:

Model File

Model files can be serialized and stored, typically as pickle files when using python.

You may end up storing several versions of your model:

- You find one you want to save for later
- You find one you want to deploy
- You create a new version to replace the previously deployed model

Artifact Tracking in MLFlow

```
Import mlflow

with mlflow.start_run():
      …code…

      mlflow.set_tag(“Tag_name”, “Tag value”)
      mlflow.log_params(params)
      mlflow.log_metric(“metric_name”, metric_value)

      mlflow.log_artifacts(“path/to/artifact”)

      mlflow.end_run()
```

What is a model registry?

- A model registry is where you store and register your models.

Model File

Models are versioned.

Model files are retrieved from the registry to deploy in production.

Data

Plots/images

MLFlow Artifact Tracking and Model Registry Demo

Experiment Tracking Lab