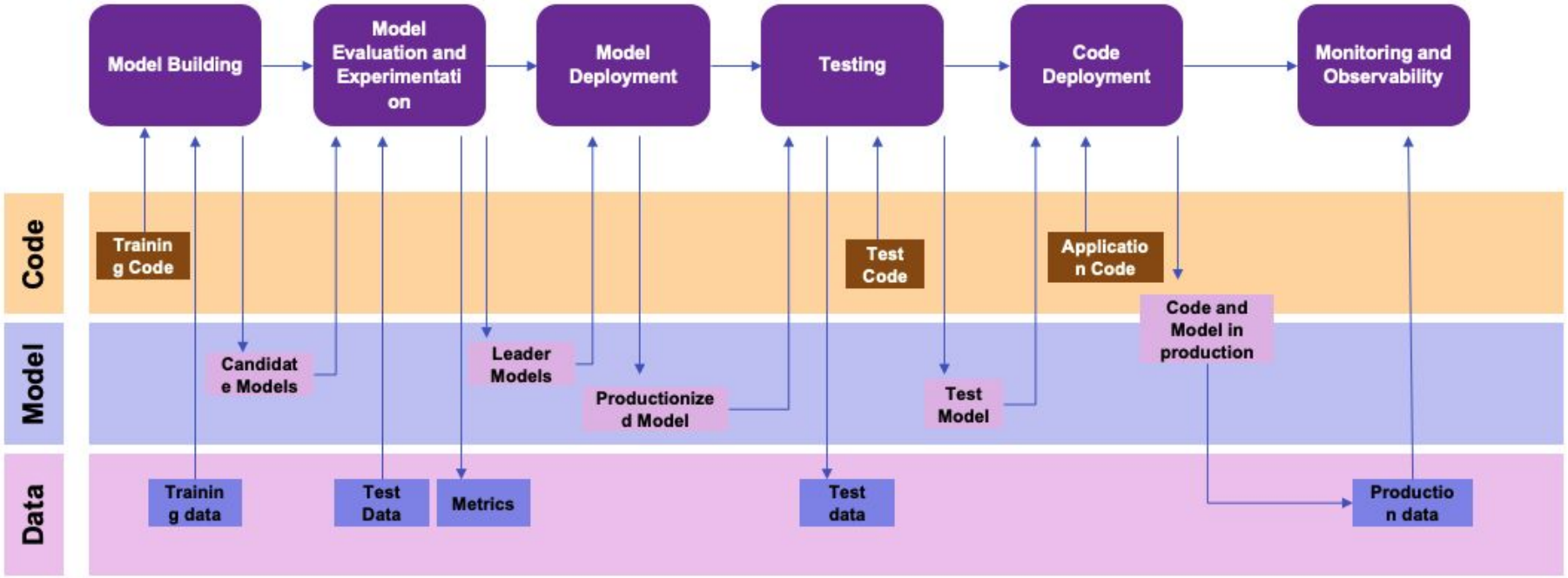
MLOps Hands-On

Part II: DVC

MLOps… how?


Model Building
Model Evaluation and Experimentation
Model Deployment
Testing
Code Deployment
Monitoring and Observability
Code
Model
Data
Training Code
Test Code
Application Code
Candidate Models
Leader Models
Productionized Model
Test Model
Code and Model in production
Training data
Test Data
Metrics
Test data
Production data

Hands-on idea and tooling

Target: to train a ML model implementing the ML lifecycle with **mlflow** and **dvc**

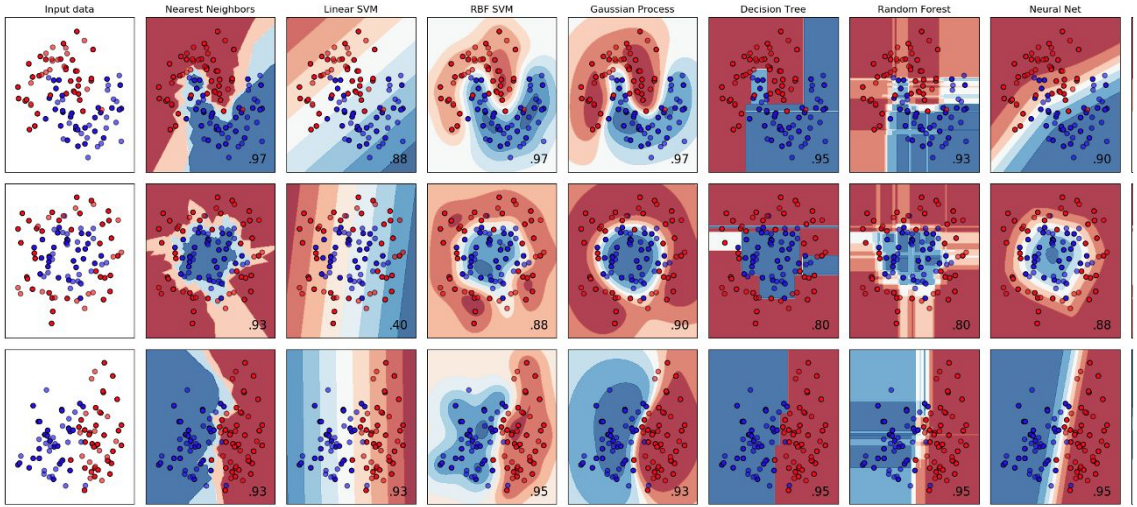
Features to accomplish:

- using python, train a model and register it
- control the model versioning
- register the model performance
- if there is any change in the code, retrain the model
- if there is any change in the data, retrain the model

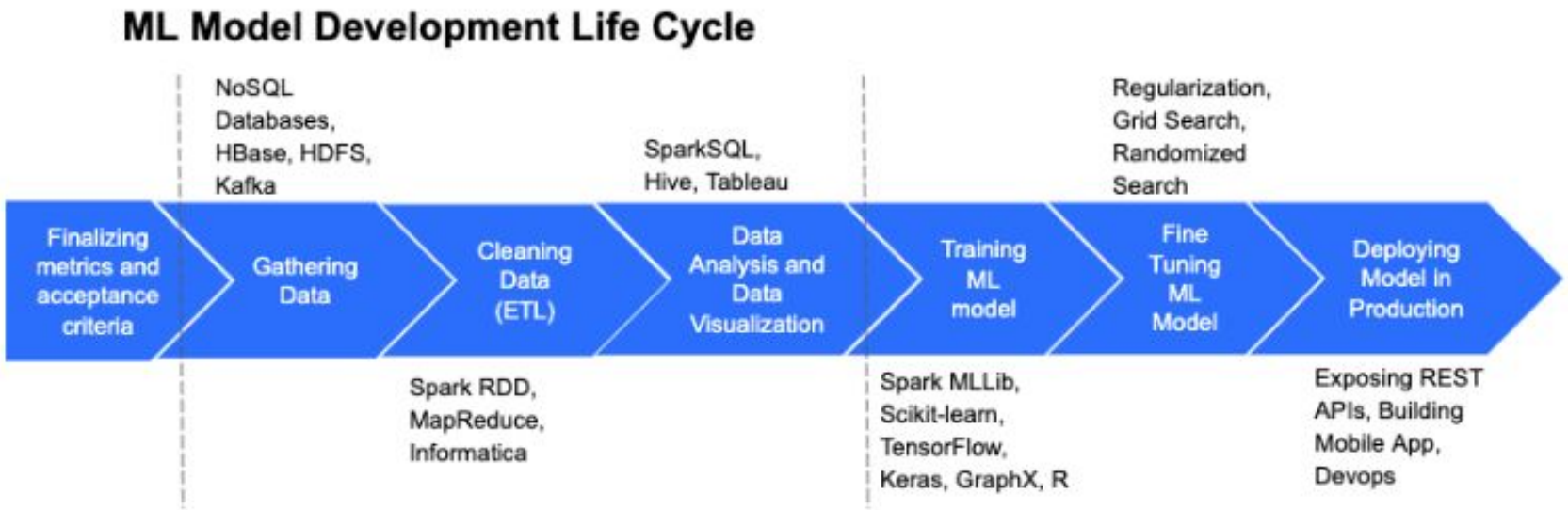
Tools and Frameworks

How can you track the models: MLFlow

- Remember, we want to track the models because…
  - Data always change
  - We want to try different algorithms and hyperparameters
  - The model is a binary digital asset, would be great to have it under control
  - .



ML Model Development Life Cycle



DVC: Data Version Control

1. Instal DVC in the course python environment: `pip install dvc`
2. Download this data and move it into the *data/external* project folder
3. **Pull** the *course git repo* and see the new folder **dvc_lab**
4. Copy the **code** (src folder) and the **.yaml files** to the ***"coockicutter"*** project (this step is like you are developing the code)

Docs

Review the code: params.yaml

dvc_lab_test_01 > ! params.yaml > {} model_monitor > monitor_dashboard_html

```yaml
external_data_config:
  external_data_csv: data/external/train.csv

raw_data_config:
  raw_data_csv: data/raw/train.csv
  model_var: ['churn','number_vmail_messages','total_day_calls','total_eve_minutes','total_eve_charge','total_intl_minutes','number_customer_service
  train_test_split_ratio: 0.2
  target: churn
  random_state: 111
  new_train_data_csv: data/raw/train_new.csv

processed_data_config:
  train_data_csv: data/processed/churn_train.csv
  test_data_csv:  data/processed/churn_test.csv

mlflow_config:
  artifacts_dir: artifacts
  experiment_name: model_iteration1
  run_name: random_forest
  registered_model_name: random_forest_model
  remote_server_uri: http://localhost:8889

random_forest:
  max_depth: 10
  n_estimators: 20

model_dir: models/model.joblib

model_webapp_dir: webapp/model_webapp_dir/model.joblib

model_monitor:
  target_col_name: target
  monitor_dashboard_html: reports/data_and_target_drift_dashboard.html
```

Review the code: data preparation, training and evaluation

dvc_lab_test_01 > ! dvc.yaml > {} stages

```yaml
stages:
  raw_dataset_creation:
    cmd: python src/data/load_data.py --config=params.yaml
    deps:
    - src/data/load_data.py
    - data/external/train.csv
    outs:
    - data/raw/train.csv

  split_data:
    cmd: python src/data/split_data.py --config=params.yaml
    deps:
    - src/data/split_data.py
    - data/raw/train.csv
    outs:
    - data/processed/churn_train.csv
    - data/processed/churn_test.csv

  model_train:
    cmd: python src/models/train_model.py --config=params.yaml
    deps:
    - data/processed/churn_train.csv
    - data/processed/churn_test.csv
    - src/models/train_model.py
    params:
    - random_forest.max_depth
    - random_forest.n_estimators

  #log_production_model:
  #  cmd: python src/models/production_model_selection.py --config=params.yaml
  #  deps:
  #  - src/models/production_model_selection.py
  #  params:
  #  - random_forest.max_depth
  #  - random_forest.n_estimators
  #  outs:
  #  - models/model.joblib
```

Demo: train a model and see the result in mlflow

1. Init DVC in the “coockicutter” folder: **dvc init**
2. add *train.csv* to the dvc: **“dvc add train.csv”**
3. run the mlflow server
4. in **other** Terminal, run “**dvc repro**”
5. review the result in mlflow
6. run again “dvc repro”
7. What happened?
8. change a line in the ***train.csv*** file
9. run again “**dvc repro**”
10. change a hyperparameter in the ***params.yaml***
11. run again “**dvc repro**”


```
mlflow server \
    --backend-store-uri sqlite:///mlflow.db \
    --default-artifact-root ./artifacts \
    --host 0.0.0.0 --port 8889
```

Lab: do it yourself

Serving the model with Flask

1) Copy the Course repo folder “**webapp**” to the “**coockiecutter**” folder
2) Copy the file app.py from the course repo to the “**coockiecutter**” folder
3) uncomment the pipeline (**dvc.yaml**) to move the model to the right place
4) run **dvc repro** (what happened?)
5) run flask server: **flask --app app run --port=5002**