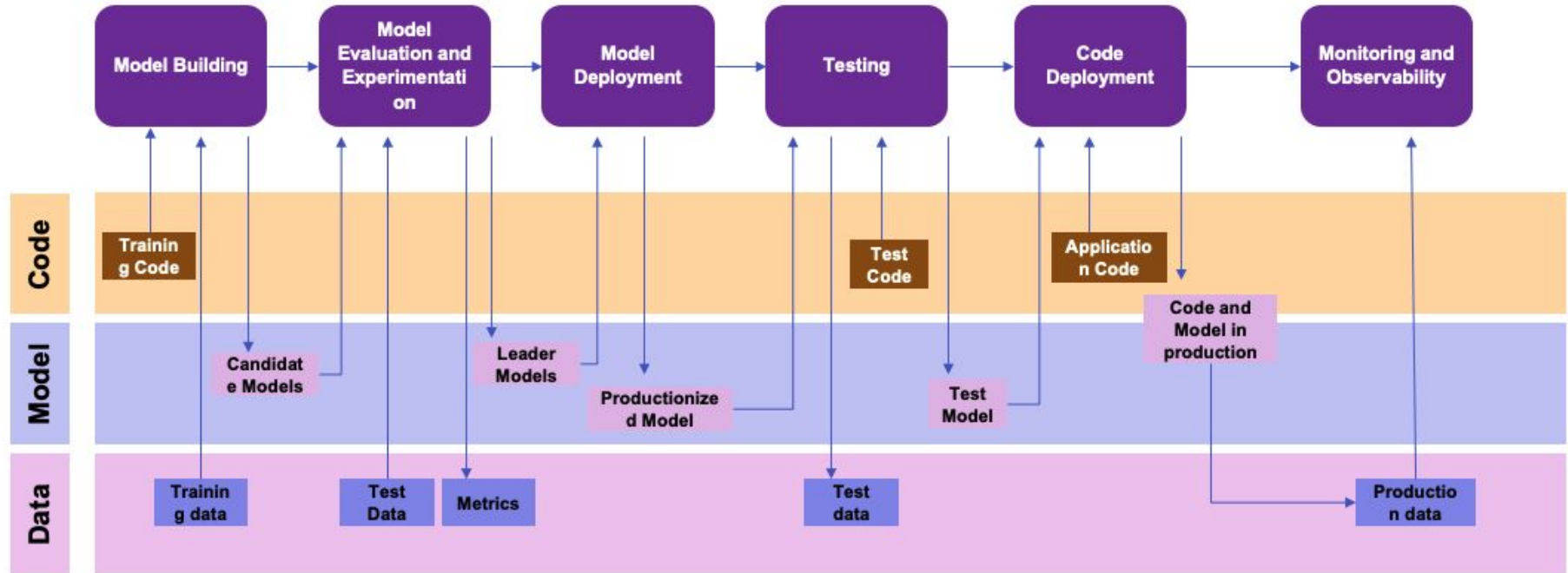


MLOps Hands-On

Part II: DVC

MLOps... how?



Hands-on idea and tooling

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

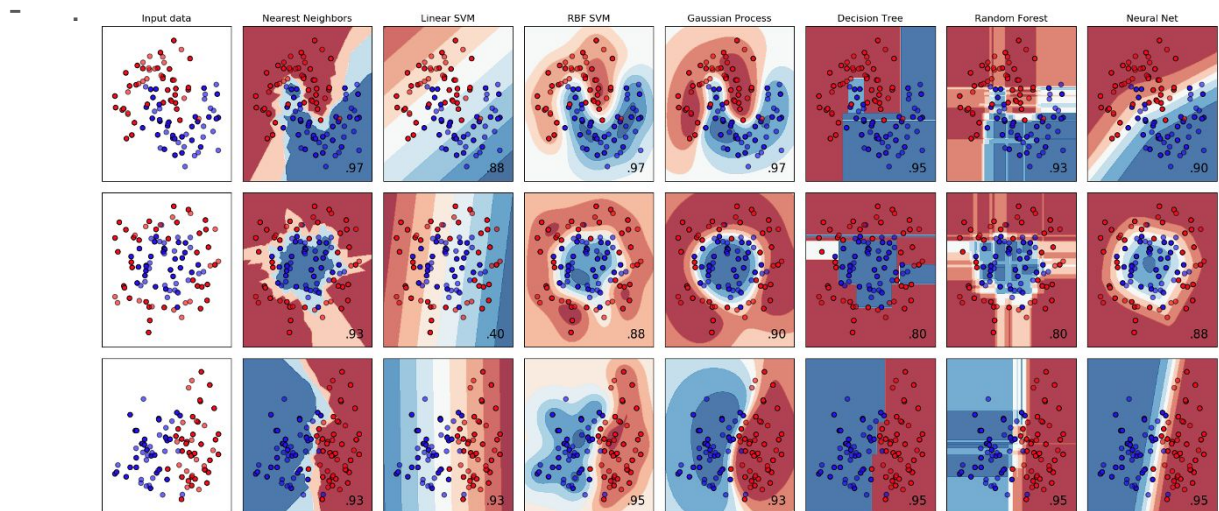
Features to accomplish:

- 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

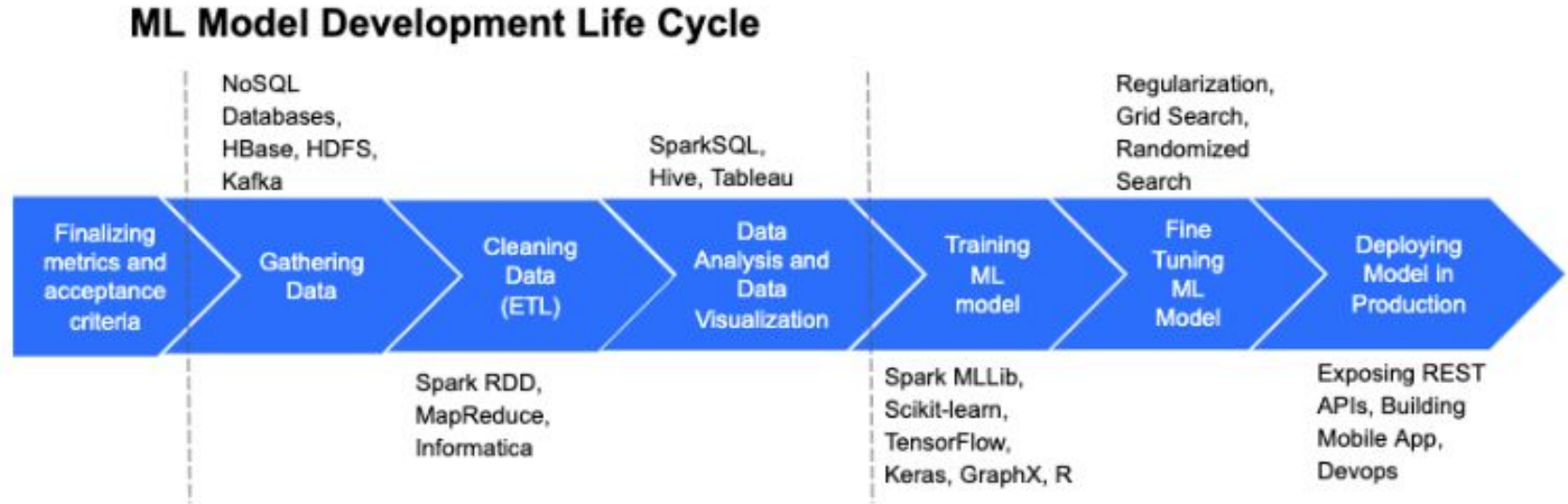
Tools and Frameworks

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)

Review the code: params.yaml


```
dvc_lab_test_01 > ! params.yaml > {} model_monitor > monitor_dashboard_html
1  external_data_config:
2    external_data_csv: data/external/train.csv
3
4  raw_data_config:
5    raw_data_csv: data/raw/train.csv
6    model_var: ['churn', 'number_vmail_messages', 'total_day_calls', 'total_eve_minutes', 'total_eve_charge', 'total_intl_minutes', 'number_customer_service
7    train_test_split_ratio: 0.2
8    target: churn
9    random_state: 111
10   new_train_data_csv: data/raw/train_new.csv
11
12  processed_data_config:
13    train_data_csv: data/processed/churn_train.csv
14    test_data_csv: data/processed/churn_test.csv
15
16  mlflow_config:
17    artifacts_dir: artifacts
18    experiment_name: model_iteration1
19    run_name: random_forest
20    registered_model_name: random_forest_model
21    remote_server_uri: http://localhost:8889
22
23  random_forest:
24    max_depth: 10
25    n_estimators: 20
26
27  model_dir: models/model.joblib
28
29  model_webapp_dir: webapp/model_webapp_dir/model.joblib
30
31  model_monitor:
32    target_col_name: target
33    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
1 stages:
2   raw_dataset_creation:
3     cmd: python src/data/load_data.py --config=params.yaml
4     deps:
5       - src/data/load_data.py
6       - data/external/train.csv
7     outs:
8       - data/raw/train.csv
9
10  split_data:
11    cmd: python src/data/split_data.py --config=params.yaml
12    deps:
13      - src/data/split_data.py
14      - data/raw/train.csv
15    outs:
16      - data/processed/churn_train.csv
17      - data/processed/churn_test.csv
18
19  model_train:
20    cmd: python src/models/train_model.py --config=params.yaml
21    deps:
22      - data/processed/churn_train.csv
23      - data/processed/churn_test.csv
24      - src/models/train_model.py
25    params:
26      - random_forest.max_depth
27      - random_forest.n_estimators
28
29  #log_production_model:
30  #   cmd: python src/models/production_model_selection.py --config=params.yaml
31  #   deps:
32  #     - src/models/production_model_selection.py
33  #   params:
34  #     - random_forest.max_depth
35  #     - random_forest.n_estimators
36  #   outs:
37  #     - 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**” into the “**coockiecutter**” folder
- 2) Copy the file `app.py` from the course repo into 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**