Digging into the Landscape of Graphs Counterfactual Explainability

Lab @ AAAI 2024

M.A. Prado-Romero - B. Prenkaj - G. Stilo

aiim

aiimlab.org

Who Are We?

Mario A. Prado-Romero
PhD. Candidate
Gran Sasso Science Institute

Bardh Prenkaj
PostDoc
Sapienza University of Rome

Giovanni Stilo
Associate Prof.
University of L'Aquila

We are part of the *Artificial Intelligence & Information Mining* (aiim - pronounced as i'm /aɪm/, and aim /eɪm/)
a collective of *Individuals* (/aɪm/) who share a common *Interest* (/eɪm/) in
Artificial Intelligence, Data Mining, and Machine Learning

Roadmap

- **Part I: The GRETEL framework (20 mins) [STILO]**
  - Introduction to the challenges of developing and evaluating GCE methods
  - Basic explanation pipeline and evaluation
  - GRETEL's design, core components, and their interaction,
  - Configuration of GRETEL

- **Part II: Many explainers, one framework (20 mins) [PRADO]**
  - Analyzing how different SoTA approaches can be implemented into GRETEL
    - Search-based
    - Heuristic-based
    - Learning-based
  - Interpreting results of SoTA GCE methods

- **Part III: Extending the framework (60 mins) [PRENKAJ]**
  - Extending the main components in custom scenarios
  - Analyzing the results to get further insights
  - Students Hands-on session using the SoBigData.it R.I.

- **Part IV: What's Next? (5 mins)**
  - Open discussion on the future trends and development in the research/industry area

SoBigData.it – Work Remotely

SOBIGDATA.it
ITALIAN RESEARCH INFRASTRUCTURE

https://sobigdata.d4science.org/group/sobigdata-gateway/explore?siteId=452129899

GitHub – Work Locally (AAAI branch)

https://github.com/aiim-research/GRETEL/tree/aaai

```
git clone -b aaai https://github.com/aiim-research/GRETEL.git
```

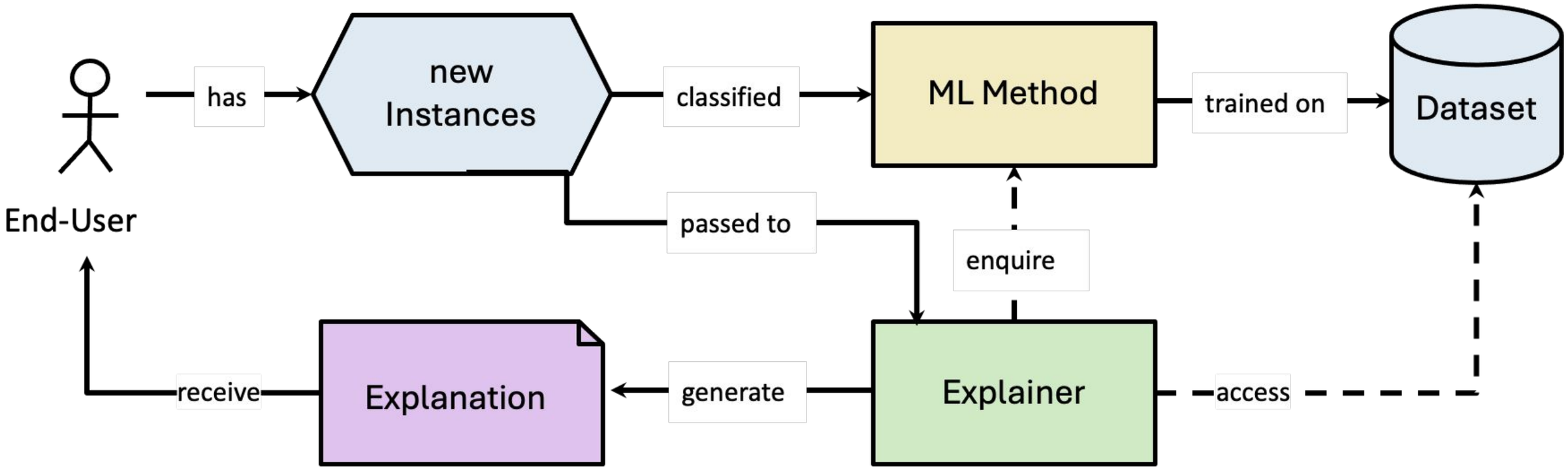
https://github.com/aiim-research/GRETEL/**wiki**#first-steps-with-gretel

Part I

The GRETEL Framework (V2)

by Giovanni Stilo

XAI workflow



GRETEL WHO?

Our goal is to create a **generic platform** that allows the researchers to **speed up** the process of **developing** *and* **testing** new **Graph Counterfactual Explanation Methods**

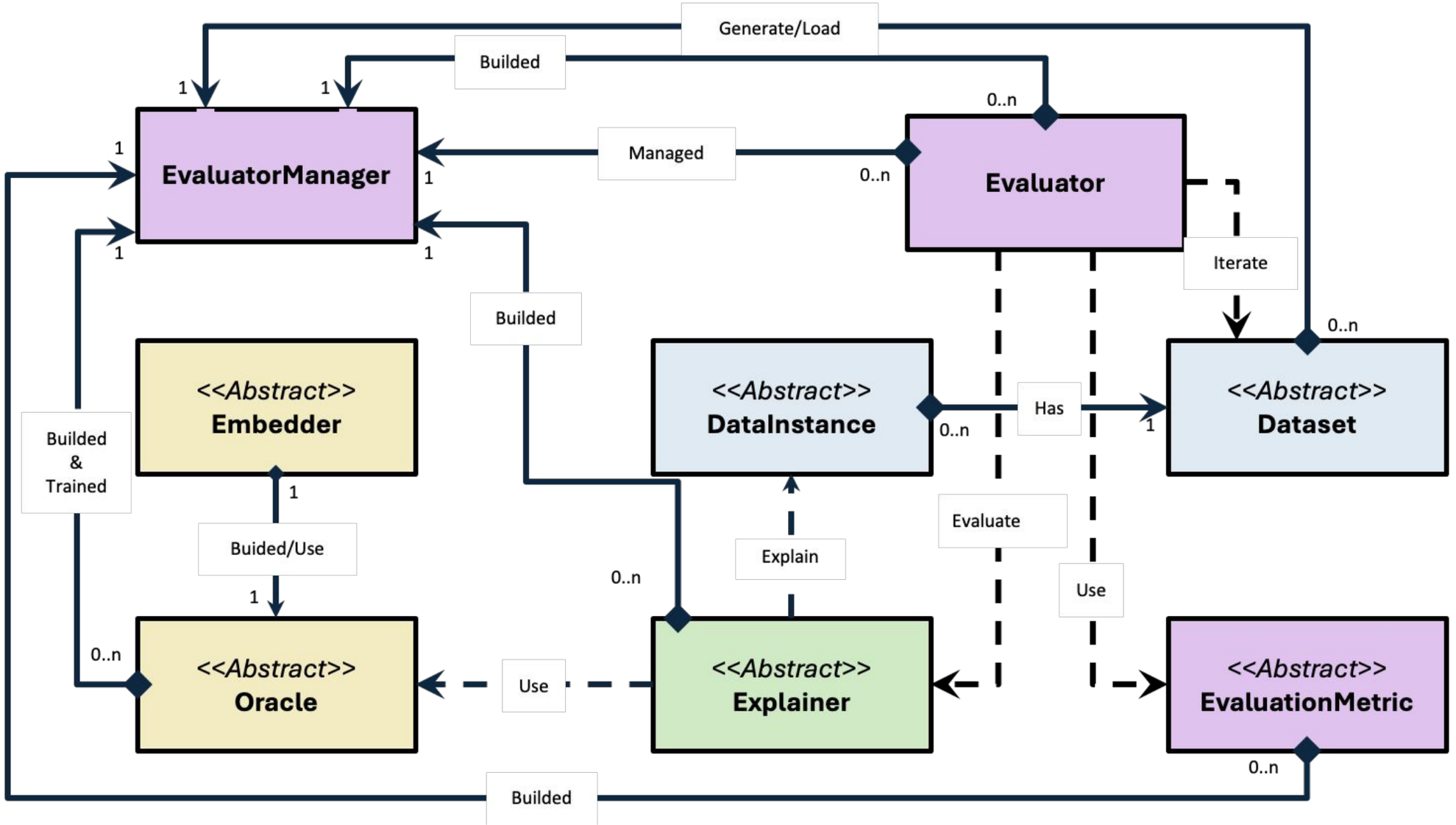
- Object Oriented paradigm;
- Inversion of Control;
- Modular + Extensible;
- Reproducibility Ready

CIKM'22 (v1)

https://github.com/aiim-research/GRETEL (v2)

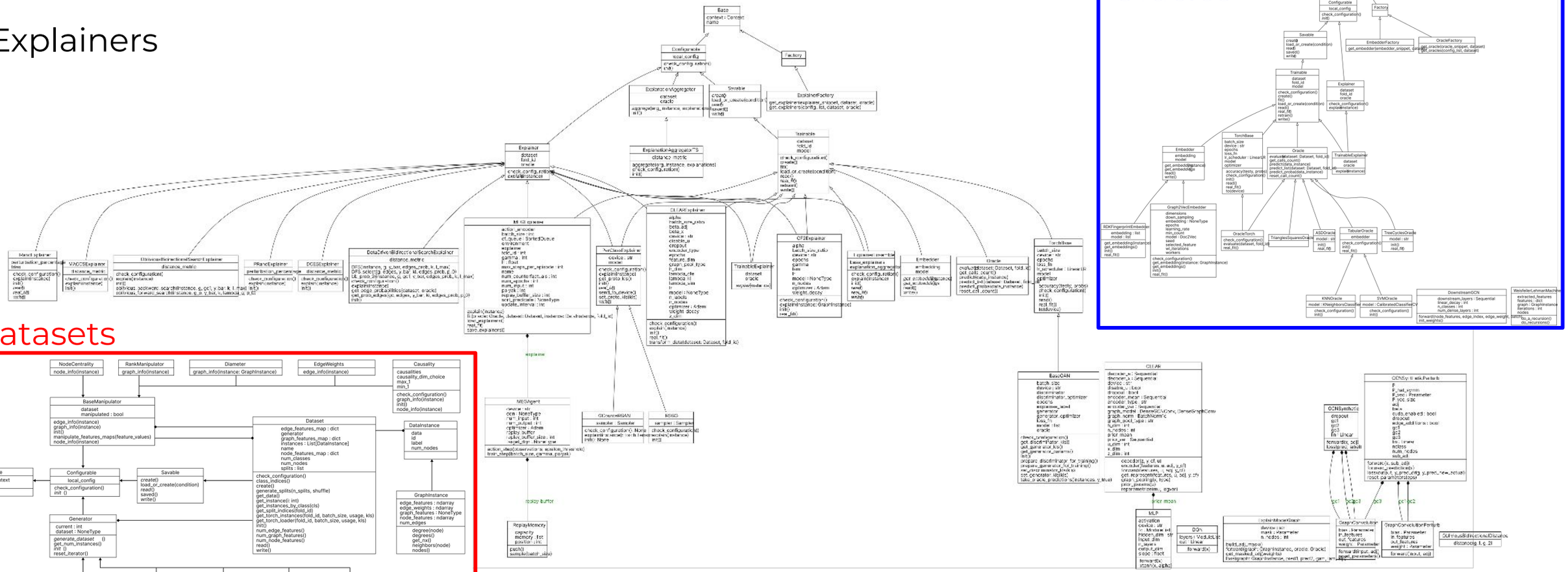
WSDM '23 (v1)

GRETEL MODULES INTERACTION



CAGING THE COMPLEXITY

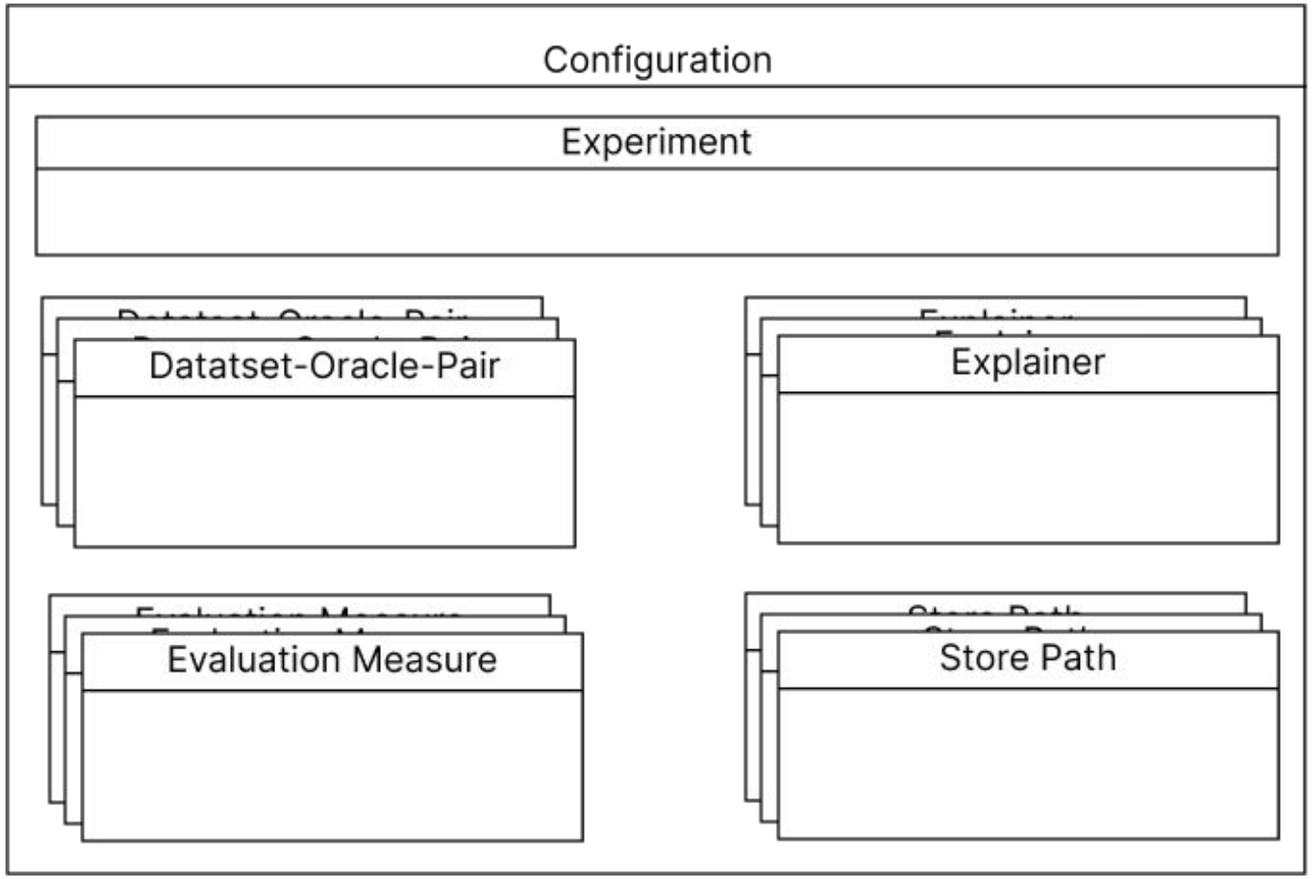
Explainers



Oracles

Datasets

Configuration Overview



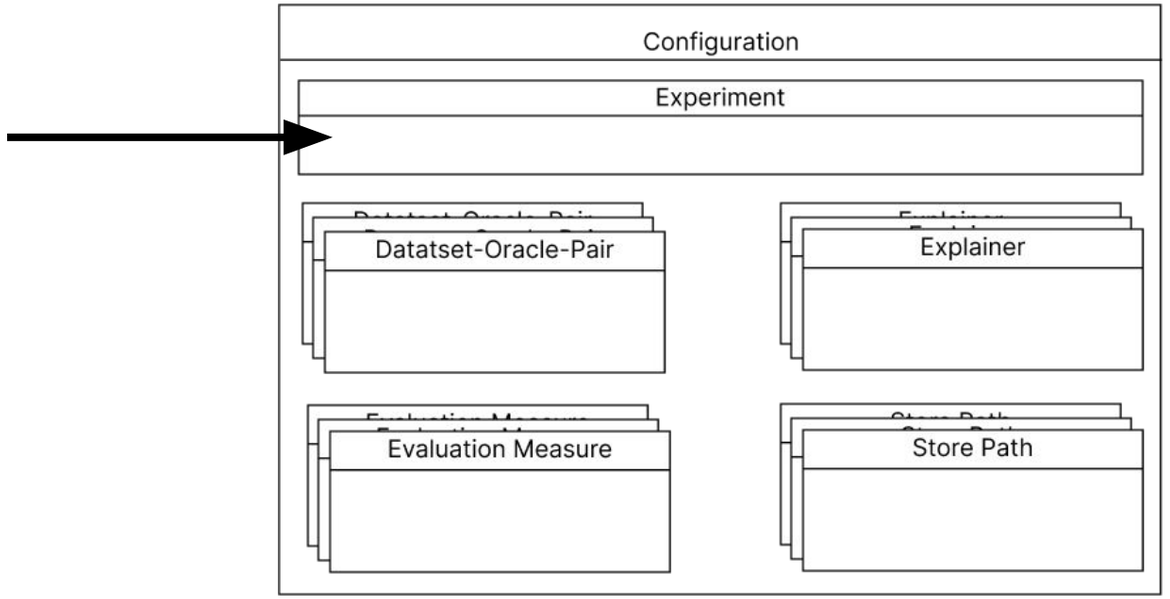

```
{
    "experiment" : {
        "scope": "examples_configs",
        "parameters" : {}
    },
    "do-pairs": [
        {"dataset" : { ... }, "oracle": { ... }},
        .
        {"dataset" : { ... }, "oracle": { ... }},
    ],
    "explainers": [
        { ... },
        .
        { ... }
    ],
    "evaluation_metrics": [
        { ... },

        .
        { ... }
    ],
    "store_paths": [
        { ... },

        .
        { ... }
    ]
}
```

Simple Object Configuation

GRETEL v2 configuration mechanism enables the configuration and the **instantiation** of the Python object directly from the **JSON snippet**.

```
"class": "src.dataset.dataset_base.Dataset",
"parameters": {
      "generator": {
                  "class": "src.dataset.generators.treecycles_rand.TreeCyclesRand",
                  "parameters": {
                        "num_instances": 128,
                        "num_nodes_per_instance": 32,
                        "ratio_nodes_in_cycles": 0.2
                   }
       }
 }
```

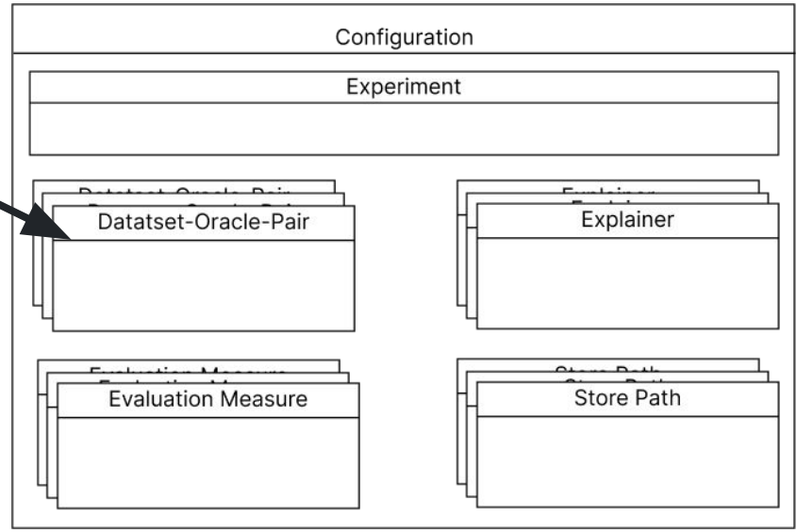
Experiment Section (& propagate)



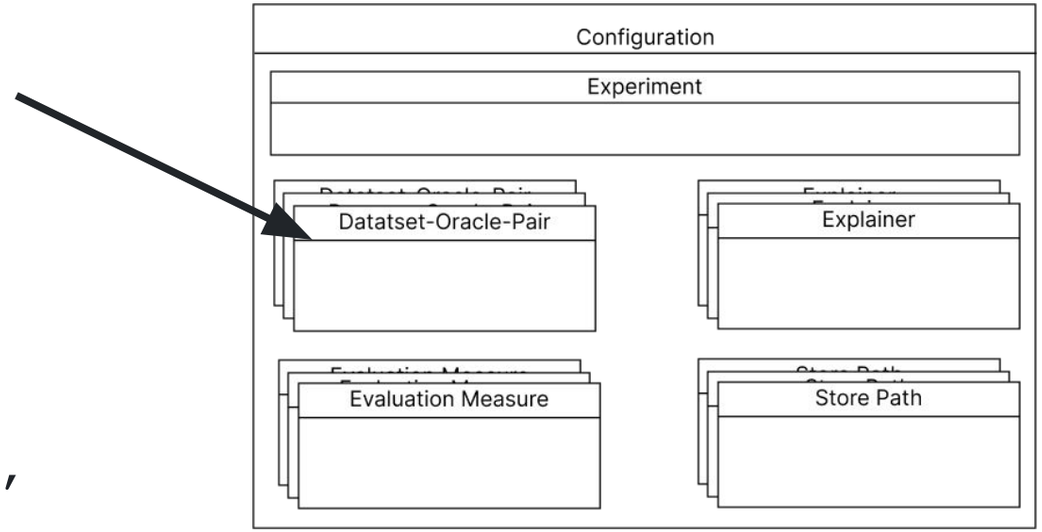

```
"experiment": {
        "scope": "examples_configs",
        "parameters": {
            "lock_release_tout":120,
            "propagate":[
                {"in_sections" : ["explainers"],"params": {"fold_id": 0}},
                {"in_sections" : ["do-pairs/oracle"],"params": {"fold_id": -1}},
            ]
        }
    },
```

DATASET

```
"dataset": {
  "class": "src.dataset.dataset_base.Dataset",
  "parameters": {
      "generator": {
        "class": "src.dataset.generators.bbbp.BBBP",
        "parameters": {
           "data_dir": "data/datasets/bbbp/",
           "data_file_name": "BBBP.csv",
           "data_label_name": "p_np"
         }
        "manipulators": [
           { "class": "src.n_dataset.manipulators.centralities.NodeCentrality", \
               "parameters": {} },
           { "class": "src.n_dataset.manipulators.weights.EdgeWeights", \
               "parameters": {} }
         ]
    }
}
```



Oracle (Simple)




```
"oracle": {
    "class": "src.oracle.tabulars.svm.SVMOracle",
    "parameters": {
        "fold_id": -1,
        "embedder": {
            "class": "src.embedder.molecule.model.RDKFingerprintEmbedder",
            "parameters": {}
        },
        "model": {  "parameters": {} }
    }
 }
```

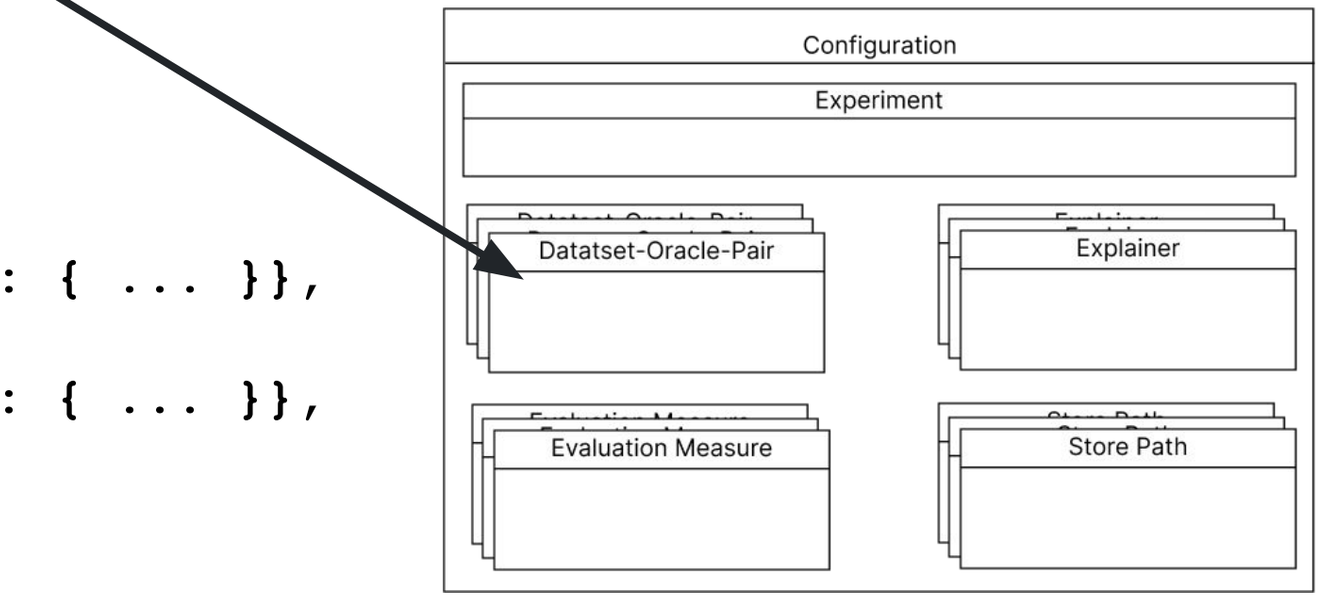
Oracle (Real)

```
"oracle": {
  "class": "src.oracle.nn.torch.OracleTorch",
  "parameters": {
    "epochs": 200,
    "batch_size": 32,
    "optimizer": {
      "class": "torch.optim.RMSprop",
      "parameters": { "lr":0.01}
    },
    "loss_fn": {
      "class": "torch.nn.CrossEntropyLoss",
      "parameters": { "reduction": "mean" }
     },
     "model": {
      "class": "src.oracle.nn.gcn.DownstreamGCN",
      "parameters":{
          "num_conv_layers":3,
          "num_dense_layers":1,
          "conv_booster":2,
          "linear_decay":1.8}
     }
  }
}
```

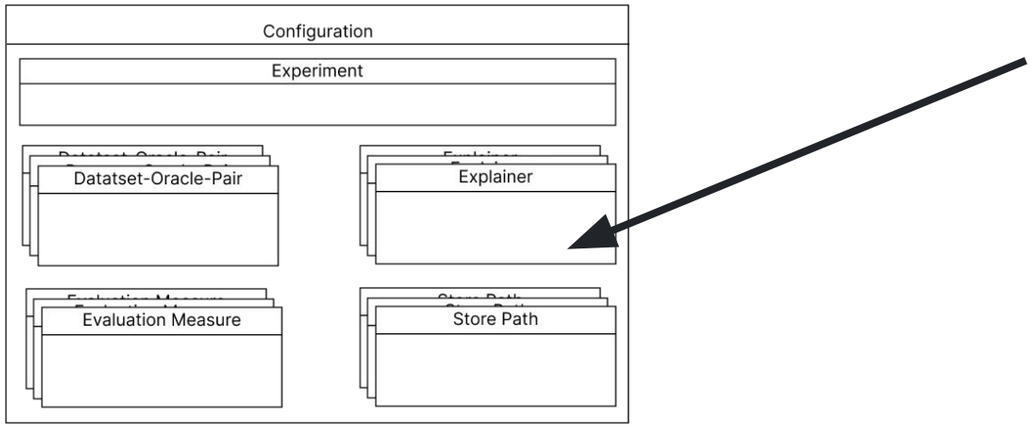
What can you note?

DO-PAIRS

```
"do-pairs": [
    {"dataset" : { ... }, "oracle": { ... }},
    .
    {"dataset" : { ... }, "oracle": { ... }},
],
```



Explainers

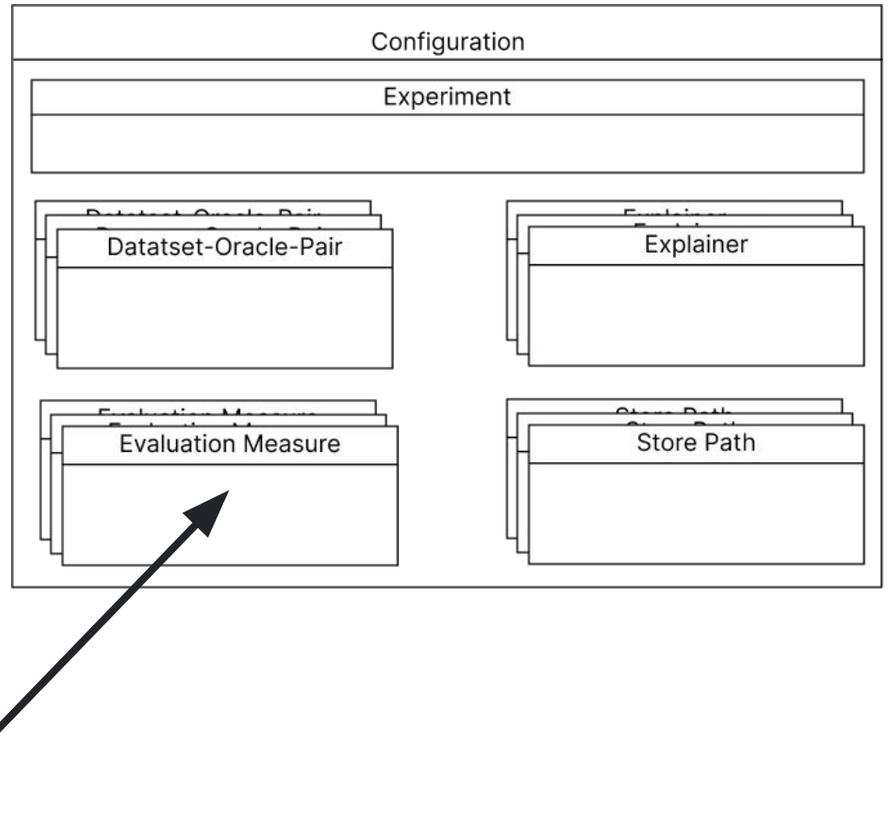



```
"explainers": [
    {"class": "src.explainer.heuristic.obs.ObliviousBidirectionalSearchExplainer",
    "parameters":{}},
    {"class": "src.explainer.search.i_rand.IRandExplainer", "parameters": {"p": 0.01, "t": 3}},
    {"class": "src.explainer.generative.cf2.CF2Explainer",
    "parameters":{"epochs": 50, "batch_size_ratio": 0.2, "lr" : 0.02, "alpha" : 0.7, "lam" : 20, \
          "gamma" : 0.9}
          },
    {"class": "src.explainer.generative.clear.CLEARExplainer",
    "parameters":{ "epochs": 100, "lr": 0.01, "lambda_cfe": 0.1, "alpha": 0.4, \
          "batch_size_ratio": 0.15 }
          },
]
```

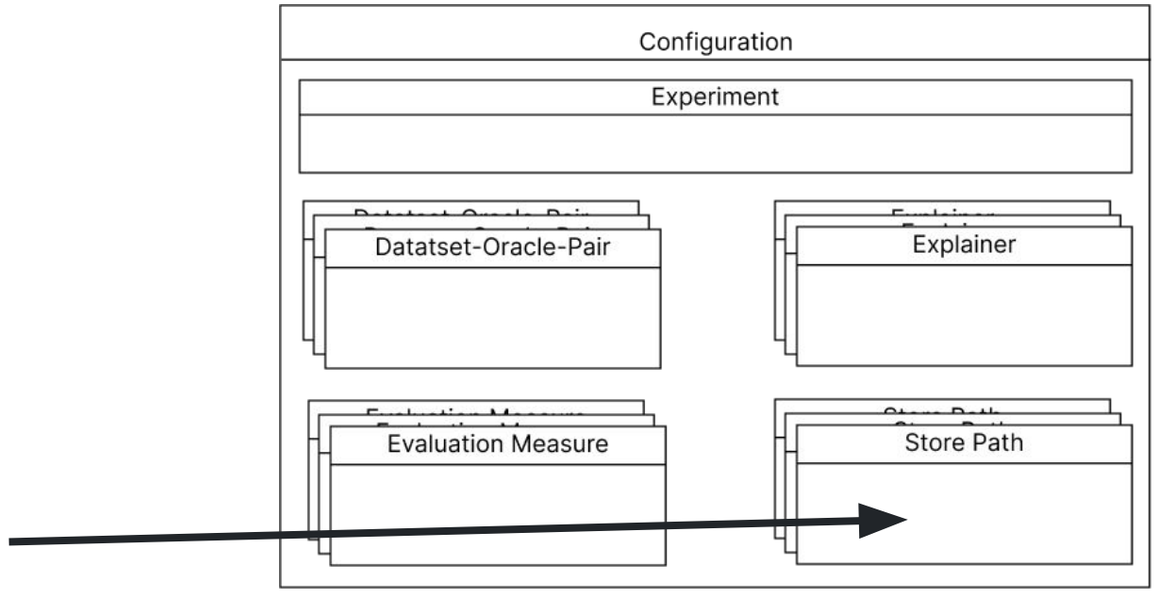
Evaluations Measure

Inherited from V1 and has not been updated yet:
not aligned with the current configuration philosophy

```
{"evaluation_metrics": [
    {"name": "runtime", "parameters": {}},
    {"name": "graph_edit_distance", "parameters": {}},
    {"name": "oracle_calls", "parameters": {}},
    {"name": "correctness", "parameters": {}},
    {"name": "sparsity", "parameters": {}},
    {"name": "fidelity", "parameters": {}},
    {"name": "oracle_accuracy", "parameters": {}}
  ]}
```



Paths & Cache




```
"store_paths": [
        {"name": "dataset_store_path", "address": "./data/cache/datasets/"},
        {"name": "oracle_store_path", "address": "./data/cache/oracles/"},
        {"name": "embedder_store_path", "address": "./data/cache/oracles/"},
        {"name": "explainer_store_path", "address": "./data/cache/explainers/"},
        {"name": "log_store_path", "address": "./output/logs/"},
        {"name": "output_store_path", "address": "./output/results/"}
    ]
```

COMPOSE (MECHANISM)

```
"compose_man": "config/snippets/datasets/centr_and_weights.json"
```

Will be replaced by the corresponding file:

```
{
    "manipulators": [
    { "class": "src.dataset.manipulators.centralities.NodeCentrality",
      "parameters": {} },
    { "class": "src.dataset.manipulators.weights.EdgeWeights",
      "parameters": {} }
    ]
 }
```

Configuration w Compose & Propagate

```
{
  "experiment": {
    "scope": "examples_configs",
    "parameters": {
        lock_release_tout":120,
        "propagate":[
            {"in_sections" : ["explainers"], "params": {"fold_id": 0}},
            {"in_sections" : ["do-pairs/oracle"], "params": {"fold_id": -1}},
            {"in_sections": ["do-pairs/dataset"], "params": { "compose_man": \
                                                    "config/snippets/datasets/centr_and_weights.json" }
        ]
    }
  },

"do-pairs":[ {"compose_bbbp_svm" : "config/snippets/do-pairs/BBBP_SVM-MOL.json"} ],
"explainers": [{"class": "src.explainer.search.dces.DCESExplainer"}],
"compose_mes" : "config/snippets/default_metrics.json",
"compose_strs" : "config/snippets/default_store_paths.json"
 }
```

Nested Compose

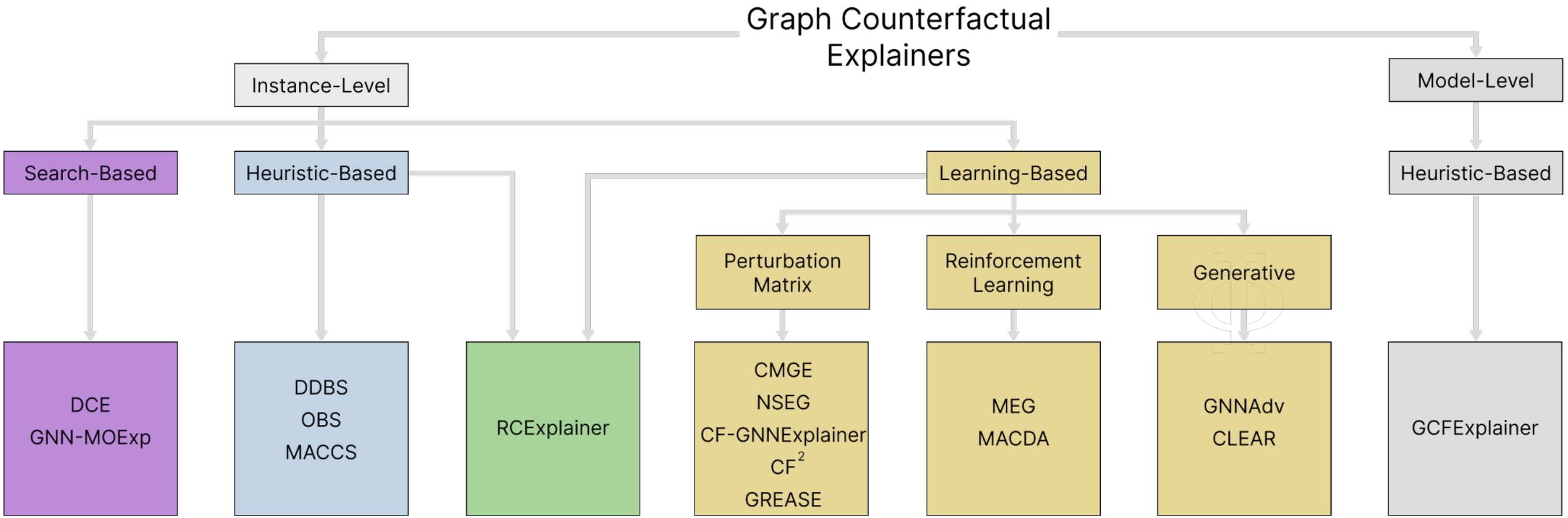
BBBP_SVM-MOL.json:

```
{
    "dataset" : {"compose_gcn" : "config/snippets/datasets/BBBP.json"},
    "oracle": {
        "class": "src.oracle.tabulars.svm.SVMOracle",
        "parameters": {
            "embedder": {
                "class": "src.embedder.molecule.model.RDKFingerprintEmbedder",
                "parameters": {}
            },
            "model": {  "parameters": {} }
        }
    }
}
```

Part II
Many Explainers One Framework

by Mario A. Prado-Romero

Taxonomy of GCE Methods



Module Structure of Gretel

- **core**
- data_analysis
- **dataset**
- embedder
- **evaluation**
- **explainer**
- **oracle**
- utils

- **core**
  - configurable
  - embedder_base
  - explainer_base
  - factory_base
  - grtl_base
  - oracle_base
  - savable
  - torch_base
  - trainable_base

- **explainer**
  - search
  - heuristic
  - rl
  - generative
  - ensemble
- **oracle**
  - custom
  - nn
  - tabulars

Oracle Base

```python
class Oracle(Trainable,metaclass=ABCMeta):
    def __init__(self, context:Context, local_config) -> None:
        super().__init__(context, local_config)
        self._call_counter = 0

    @final
    def predict(self, data_instance):...

    @final
    def predict_proba(self, data_instance):...

    @final
    def get_calls_count(self):...

    @final
    def reset_call_count(self):...

    @final
    def predict_list(self, dataset: Dataset, fold_id=0):...

    @abstractmethod
    def _real_predict(self, data_instance):
        pass

    @abstractmethod
    def _real_predict_proba(self, data_instance):
        pass
```

Example config - Default

```
{
    "experiment" : {
        "scope": "aaai_lab",
        "parameters" : {
            "lock_release_tout":120,
            "propagate":[
                {"in_sections" : ["explainers"],"params" : {"fold_id": 0}},
                {"in_sections" : ["do-pairs/oracle"],"params" : {"fold_id": -1}},
                {"in_sections": ["do-pairs/dataset"],
                 "params": { "compose_man" : "config/snippets/datasets/centr_and_weights.json" }
                }
            ]
        }
    },
    "do-pairs":[ {"compose_tcr_gcn" : "config/snippets/do-pairs/TCR-128-32-0.2_GCN.json"}],

    "explainers": [{"class": "src.explainer.generative.rsgg.RSGG","parameters":{"epochs": 150}}],

    "compose_mes" : "config/snippets/default_metrics.json",
    "compose_strs" : "config/snippets/default_store_paths.json"
}
```

Example config - Custom


```
{
    "experiment" : {
        "scope": "examples_configs",
        "parameters" : {
            "lock_release_tout": 120,
            "propagate":[
                {"in_sections" : ["explainers"],"params" : {"fold_id": 0}},
                {"in_sections" : ["do-pairs/oracle"],"params" : {"fold_id": -1,"retrain": false}}
            ]
        }
    },
    "do-pairs":[ {
        "dataset" : {
          "class": "src.dataset.dataset_base.Dataset",
          "parameters": {
              "generator": {
                  "class": "src.dataset.generators.treecycles_rand.TreeCyclesRand",
                  "parameters": { "num_instances": 150,
                                  "num_nodes_per_instance": 100,
                                  "ratio_nodes_in_cycles": 0.3 }
              }
          }
        },
        "oracle": {
          "class": "src.oracle.custom.oracle_tree_cycles.TreeCyclesOracle",
          "parameters": {}
        }
      }
    ],
    "explainers" : [{"class": "src.explainer.search.i_rand.IRandExplainer",
                     "parameters": {"p": 0.01, "t": 3}}],
    "compose_mes" : "config/snippets/default_metrics.json",
    "compose_strs" : "config/snippets/default_store_paths.json"
}
```

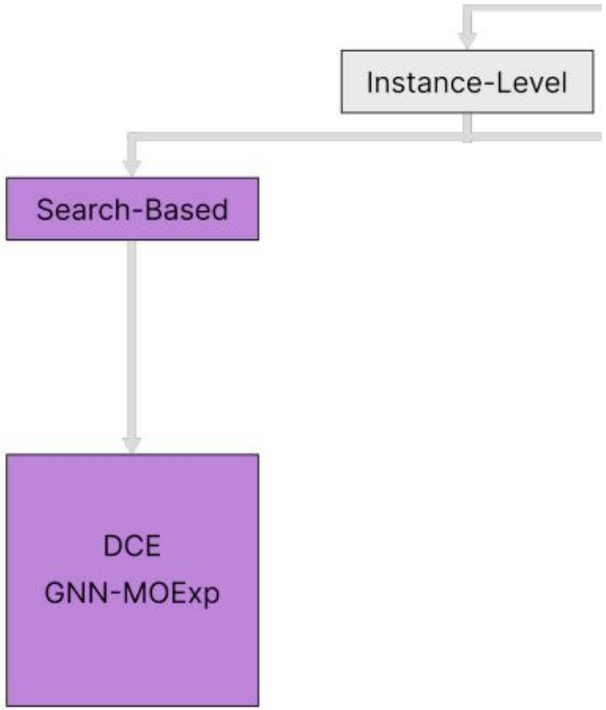

Explainer Base

```
giovanni, last month | 1 author (giovanni)
class Explainer(Configurable, metaclass=ABCMeta):

    def __init__(self, context: Context, local_config):
        self.dataset = retake_dataset(local_config)
        self.oracle = retake_oracle(local_config)
        super().__init__(context, local_config)


    @abstractmethod
    def explain(self, instance):
        pass

    def check_configuration(self):
        super().check_configuration()
        self.local_config['parameters']['fold_id'] =  self.local_config['parameters'].get('fold_id', -1)
        self.fold_id = self.local_config['parameters']['fold_id']
```

Provides access to the oracle and the training dataset

Retrieves the fold id

Search-Based Explanation Methods



- Find a counterfactual **within the data**
- For a graph $\boldsymbol{G} \epsilon \mathcal{G}$ find a $\boldsymbol{G'} \epsilon \mathcal{G}$ s.t. $\boldsymbol{\Phi}(\boldsymbol{G}) \neq \boldsymbol{\Phi}(\boldsymbol{G'})$

Search-Based Explanation Methods

```python
class DCESExplainer(Explainer):
    """The Distribution Compliant Explanation Search Explainer performs a search of
    the minimum counterfactual instance in the original dataset instead of generating
    a new instance"""

    def check_configuration(self):…


    def init(self):…

    def explain(self, instance):
        input_label = self.oracle.predict(instance)

        # if the method does not find a counterfactual example returns the original graph
        min_ctf = instance

        # Iterating over all the instances of the dataset
        min_ctf_dist = sys.float_info.max
        for ctf_candidate in self.dataset.instances:
            candidate_label = self.oracle.predict(ctf_candidate)

            if input_label != candidate_label:
                ctf_distance = self.distance_metric.evaluate(instance, ctf_candidate, self.oracle)

                if ctf_distance < min_ctf_dist:
                    min_ctf_dist = ctf_distance
                    min_ctf = ctf_candidate

        result = copy.deepcopy(min_ctf)
        result.id = instance.id

        return result
```

Find a counterfactual **within the data**

For a graph $\boldsymbol{G} \in \mathcal{G}$ find a $\boldsymbol{G'} \in \mathcal{G}$

s.t. $\boldsymbol{\Phi}(\boldsymbol{G}) \neq \boldsymbol{\Phi}(\boldsymbol{G'})$

Heuristic-Based Explainers - OBS

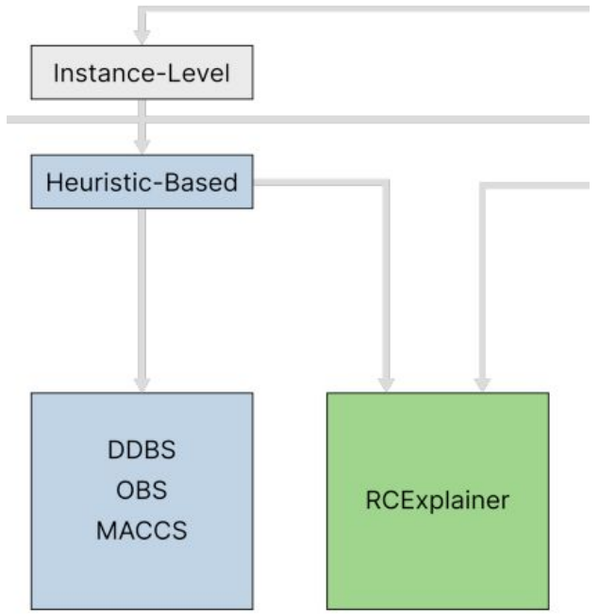
```
            gci[j][i] = 0
        else:
            gci[i][j] = 1
            gci[j][i] = 1

inst = GraphInstance(id=instance.id,
                     label=0,
                     data=gci,
                     node_features=instance.node_features)

r = self.oracle.predict(inst)

if r==y_bar:
```

gci is a numpy array used to represent the graph by OBS

Calling the oracle from inside the explainer's source code

Heuristic-Based Explainers - MACCS

```python
def _oracle_wrapper_creator(self, oracle: Oracle, dataset: Dataset):
    """
    This function takes an oracle and return a function that takes the smiles
    of a molecule, transforms it into a DataInstance and returns the prediction
    of the oracle for it
    """

    # The inner function uses the oracle, but does not receive it as a parameter
    def _oracle_wrapper(molecule_smiles):
        _ , inst = mol_gen.smile2graph(-1, molecule_smiles, 0, dataset)
        return oracle.predict(inst)

    return _oracle_wrapper
```



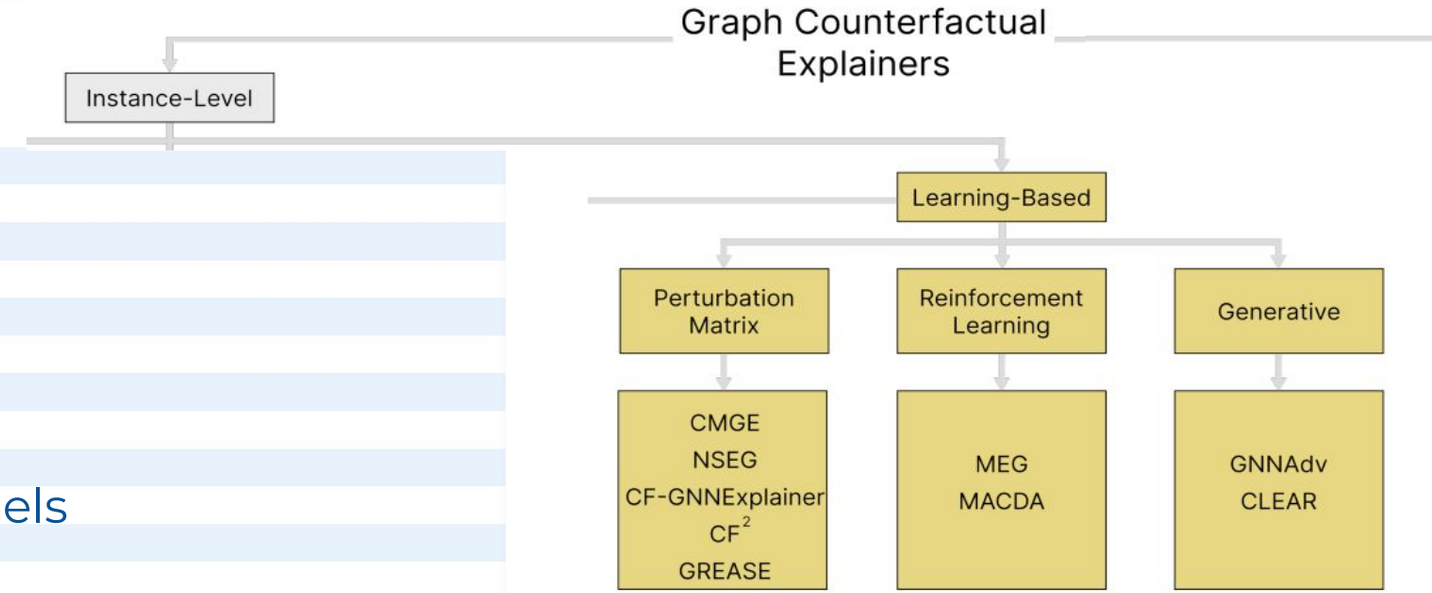

```python
def explain(self, instance):

    smiles = instance.graph_features['smile']
    clf = self._oracle_wrapper_creator(self.oracle, self.dataset)

    basic = exmol.get_basic_alphabet()
    stoned_kwargs = {"num_samples": 1500, "alphabet": basic, "max_mutations": 2}

    try:
        samples = exmol.sample_space(smiles, clf, batched=False, use_selfies=False,
        stoned_kwargs=stoned_kwargs, quiet=True)

        cfs = exmol.cf_explain(samples)
    except Exception as err:
        print('instance id:', str(instance.id))
        print(instance.graph_features['smile'])
        print(err.args)
        return instance

    if(len(cfs) > 1):
        min_cft_label = clf(cfs[1].smiles)
        _ , min_counterfactual = mol_gen.smile2graph(instance.id,
                                                     cfs[1].smiles,
                                                     min_cft_label,
                                                     self.dataset)
        return min_counterfactual
    else:
        return instance
```

Learning-Based Methods - Trainable Class

```python
class Trainable(Savable,metaclass=ABCMeta):

    def __init__(self, context: Context, local_config):...

    def load_or_create(self, condition=False):
        super().load_or_create(self._to_retrain() or condition)

    def _to_retrain(self):...

    def retrain(self):...

    def fit(self):...

    def create(self):...

    def write(self):...

    def read(self):...

    @abstractmethod
    def real_fit(self):...

    def check_configuration(self):
        super().check_configuration()
        self.local_config['parameters']['fold_id'] =  self.local_config['parameters'].get('fold_id', -1)
        self.fold_id = self.local_config['parameters']['fold_id']
```

saving/loading trained models

Retrieves the fold id



Using the Evaluator Manager

```python
config_path = os.path.join(module_path, 'lab', 'config', config_f_name)
runno = 1

print(f"Generating context for: {config_path}")
context = Context.get_context(config_path)
context.run_number = runno

context.logger.info(f"Executing: {context.config_file} Run: {context.run_number}")
context.logger.info("Creating the evaluation manager.....................................")


if 'do-pairs' in context.conf:
    context.logger.info(f"Creating the paired  evaluators.....................................")
    eval_manager = PairedEvaluatorManager(context)
else:
    context.logger.info("Creating the evaluators.....................................................
    eval_manager = EvaluatorManager(context)

context.logger.info(
    "Evaluating the explainers..............................................................."
)
eval_manager.evaluate()
```

Python

Analyzing the Counterfactuals

```python
evaluator = eval_manager.evaluators[0]
evaluator
```

Python

```python
inst_cf_pairs = evaluator.get_instance_explanation_pairs()
og_inst = inst_cf_pairs[6][0]
cf_inst = inst_cf_pairs[6][1]
```

Python

```python
from src.data_analysis.data_analyzer import DataAnalyzer as dan
changes = dan.get_cf_changes(og_inst, cf_inst, False)
```

Python

```
added edges: [(7, 28), (8, 29)]
removed_edges: []
added nodes: []
removed nodes: []
```

Visualizing the Counterfactuals

```python
import networkx as nx
import matplotlib.pyplot as plt
%matplotlib inline
```

Python

```python
layout = nx.spring_layout
pos = layout(og_inst.get_nx())
```

Python

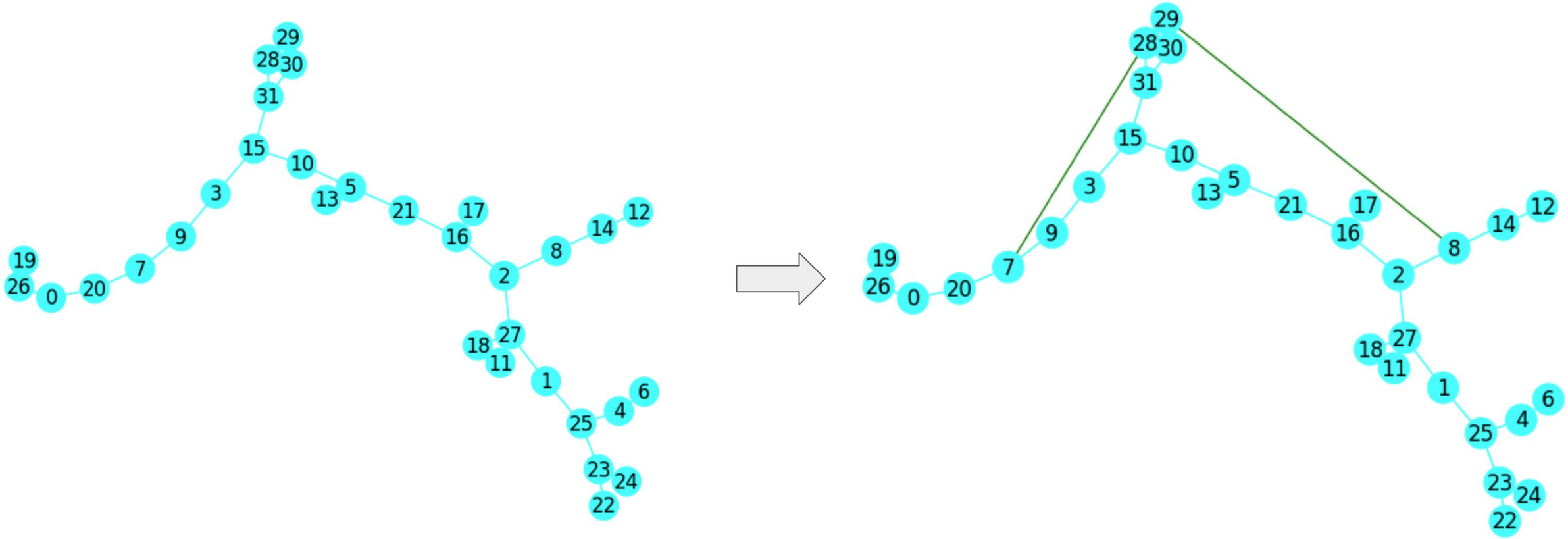
```python
pos = dan.draw_graph(og_inst)
```

Python

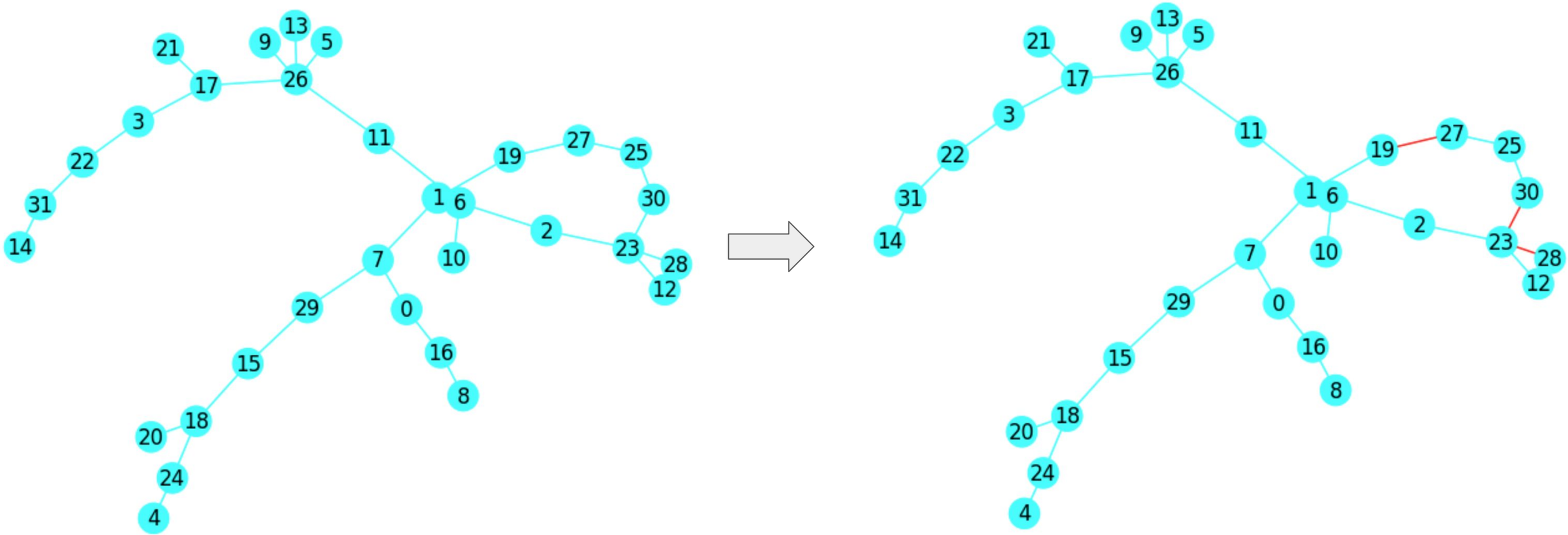
```python
dan.draw_counterfactual_actions(og_inst, cf_inst, position=pos)
```

Python

Creating a Cycle



Breaking a Cycle



Part III
Extending the Framework (Live Demo)

by Bardh Prenkaj

GitHub – Work Locally (AAAI branch)

https://github.com/aiim-research/GRETEL/tree/aaai

GitHub

```
git clone -b aaai https://github.com/aiim-research/GRETEL.git
```

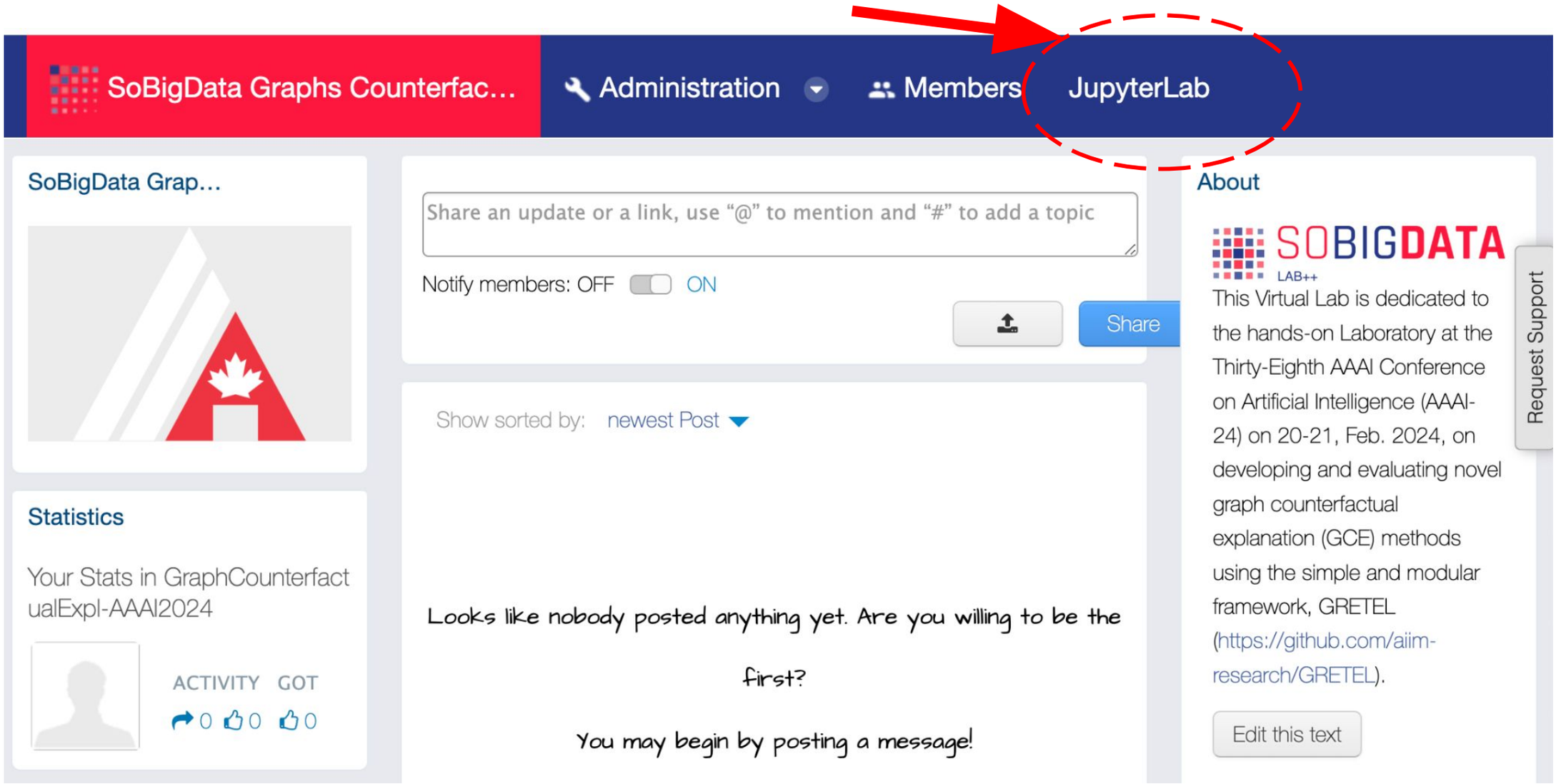
https://github.com/aiim-research/GRETEL/wiki#first-steps-with-gretel

SoBigData.it – Work Remotely

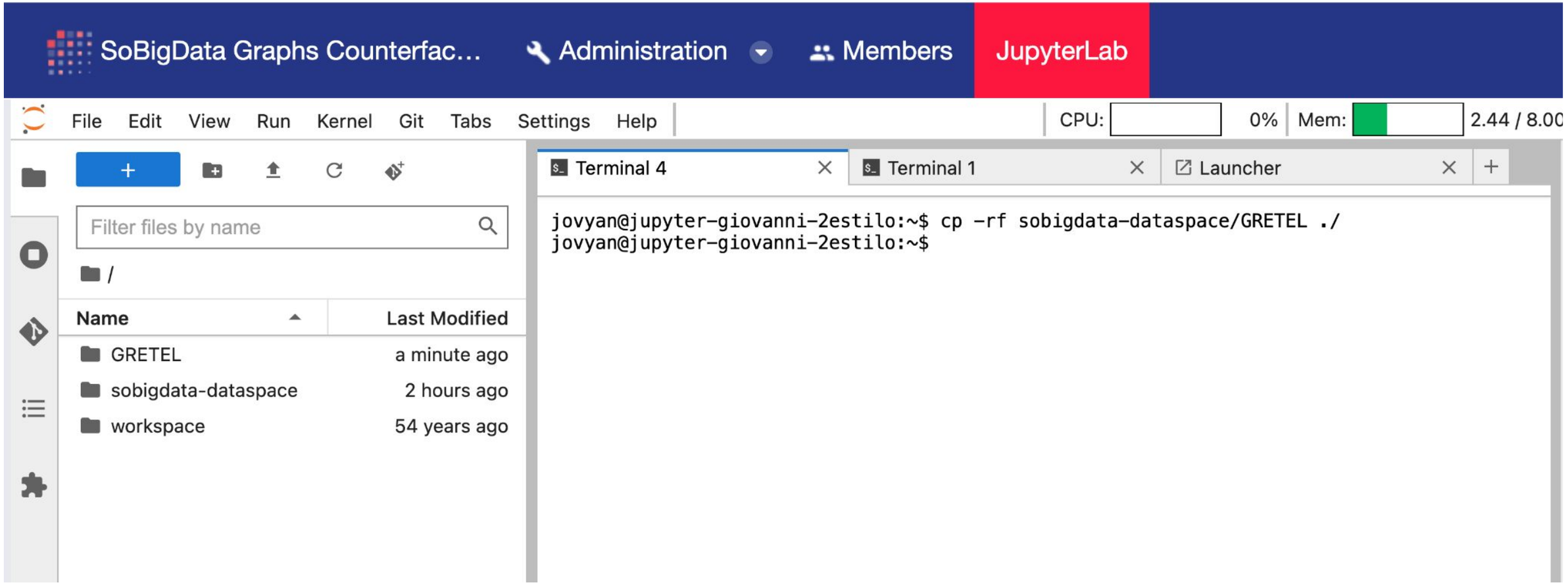
SOBIGDATA.it
ITALIAN RESEARCH INFRASTRUCTURE

https://sobigdata.d4science.org/group/sobigdata-gateway/explore?siteId=452129899

JupyterLab

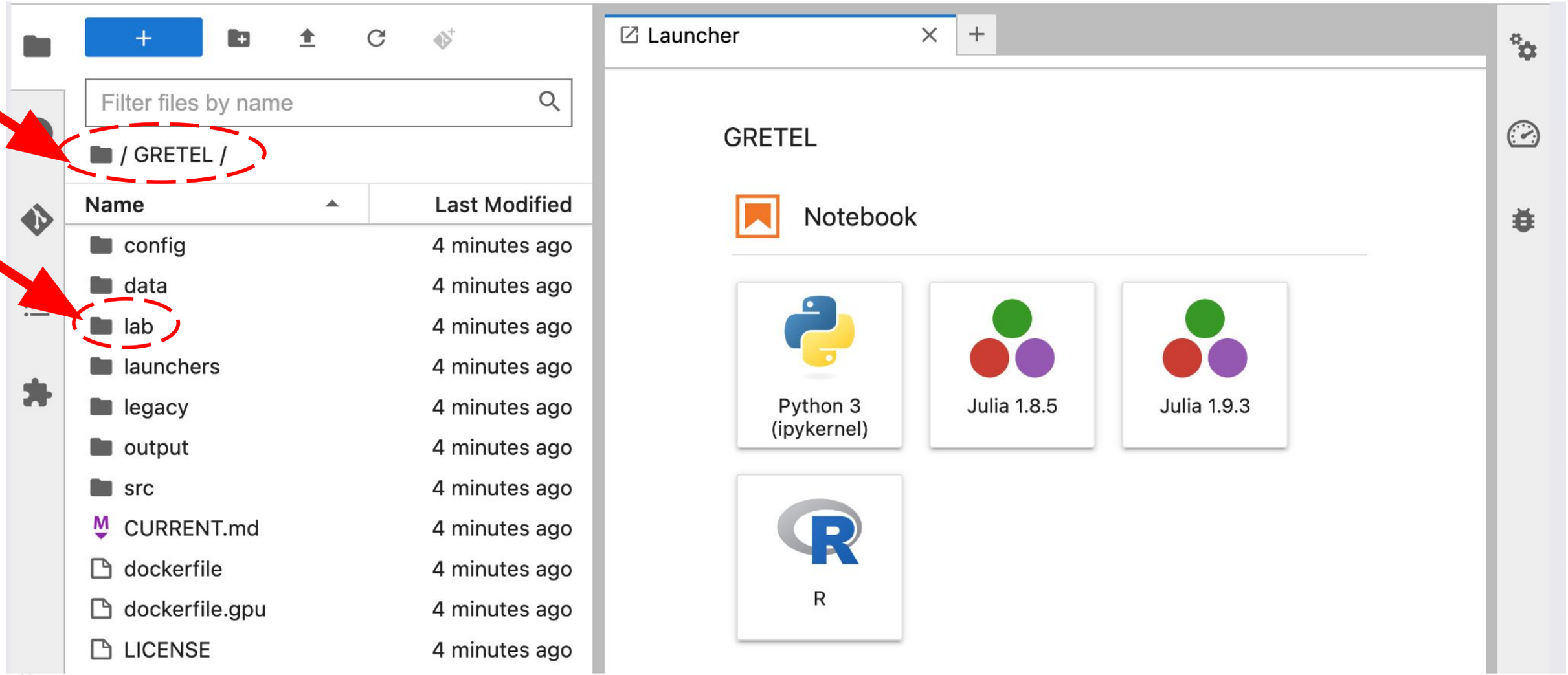

SoBigData Graphs Counterfac…
Administration
Members
JupyterLab
SoBigData Grap…
Share an update or a link, use "@" to mention and "#" to add a topic
Notify members: OFF ON
Share
Statistics
Your Stats in GraphCounterfactualExpl-AAAI2024
ACTIVITY GOT
0 0 0
Show sorted by: newest Post
Looks like nobody posted anything yet. Are you willing to be the first?
You may begin by posting a message!
About
SOBIGDATA LAB++
This Virtual Lab is dedicated to the hands-on Laboratory at the Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI-24) on 20-21, Feb. 2024, on developing and evaluating novel graph counterfactual explanation (GCE) methods using the simple and modular framework, GRETEL (https://github.com/aiim-research/GRETEL).
Edit this text
Request Support

Prepare Your Working Directory

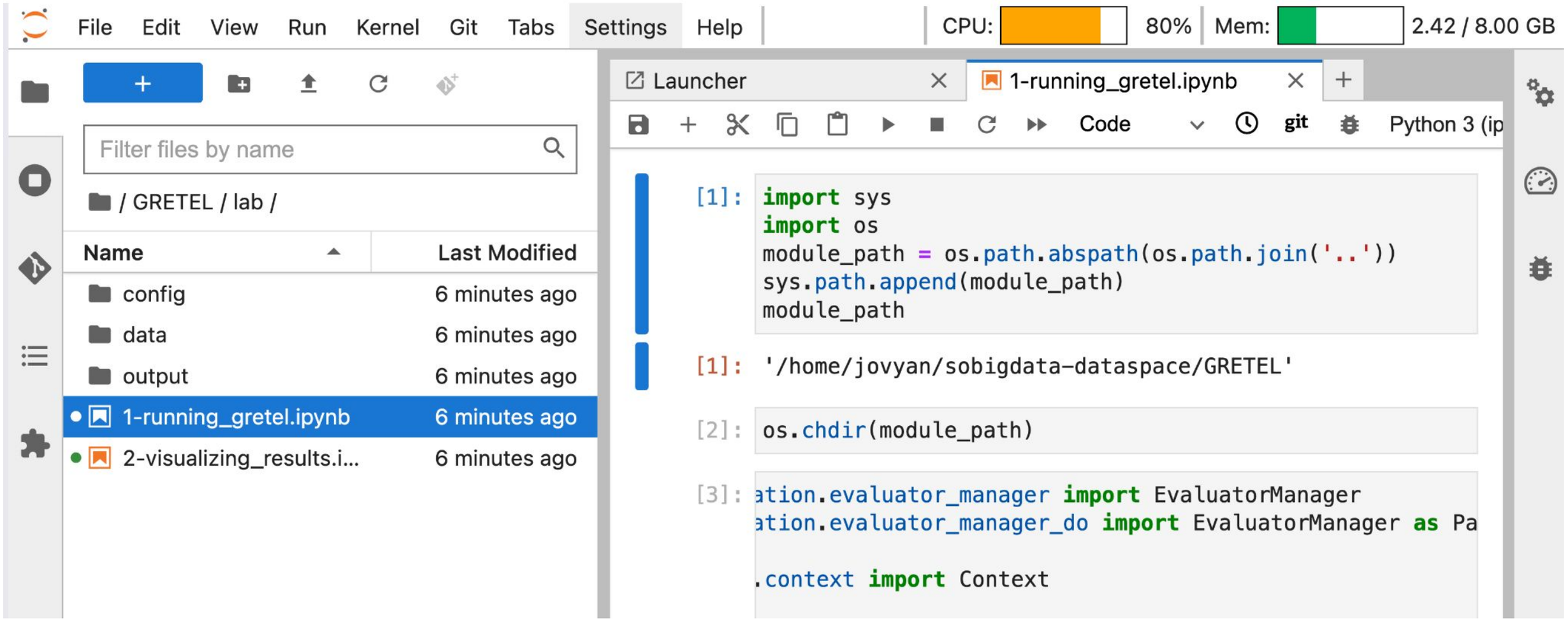


cp -rf sobigdata-dataspace/GRETEL ./

GRETEL @ SoBigData.it



GRETEL LAB


File Edit View Run Kernel Git Tabs Settings Help
CPU: 80% Mem: 2.42 / 8.00 GB
Filter files by name
/ GRETEL / lab /
Name
Last Modified
config 6 minutes ago
data 6 minutes ago
output 6 minutes ago
1-running_gretel.ipynb 6 minutes ago
2-visualizing_results.i... 6 minutes ago
Launcher
1-running_gretel.ipynb
Code
git
Python 3 (ip
[1]: import sys
import os
module_path = os.path.abspath(os.path.join('..'))
sys.path.append(module_path)
module_path
[1]: '/home/jovyan/sobigdata-dataspace/GRETEL'
[2]: os.chdir(module_path)
[3]: ation.evaluator_manager import EvaluatorManager
ation.evaluator_manager_do import EvaluatorManager as Pa
.context import Context

Part IV
What's Next?

What's Next

- **Supporting more types of learning tasks**
  - Node Classification
  - Link Prediction
- **Providing a more flexible Explanation object**
  - Multiple Explanations
  - Global Explanations
- **Including Factual Explanation Methods**
- **Providing Explanation Ensembles**

Thanks for your attention!

FOR QUESTIONS AND DISCUSSION

WE MEET AT 15:30 IN FRONT OF 121.

RSVP TO WHOVA

https://github.com/aiim-research/GRETEL

Mario A. Prado-Romero
marioalfonso.prado@gssi.it

Bardh Prenkaj
prenkaj@di.uniroma1.it

Giovanni Stilo
giovanni.stilo@univaq.it