Energy Innovation Summit
Modeling Meets Optimization

Model Building with gurobipy

Jaromił Najman
Senior Developer

Agenda

Model Building Frameworks

gurobipy, Pandas-, and Matrix-API

Integrating Machine Learning

Model Building Frameworks

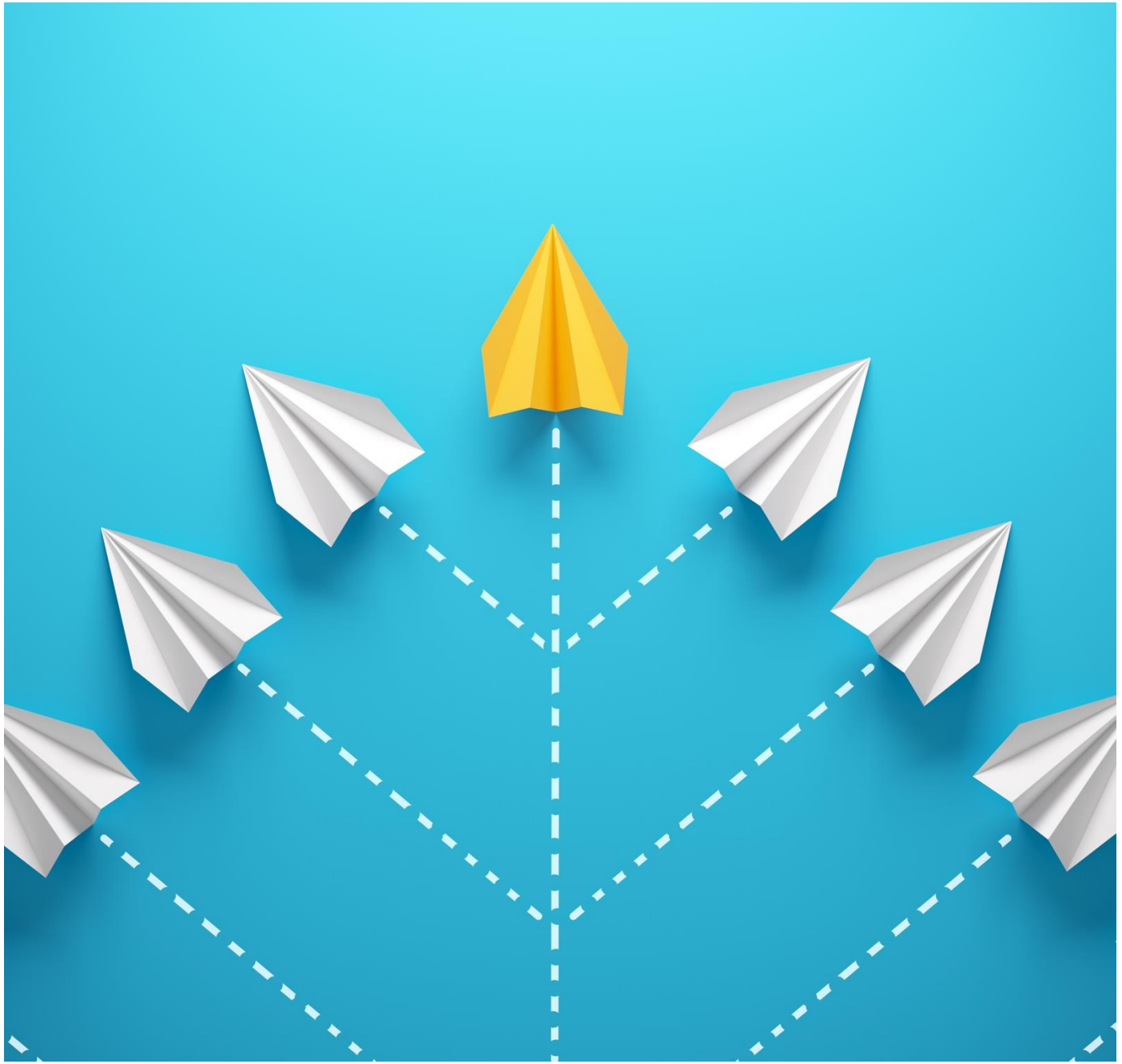
GUROBI OPTIMIZATION

Making a choice

Energy model generators
PyPSA, oemof, GenX, etc.

Third-party model-building frameworks
linopy, JuMP, PuLP, Pyomo, etc.

Gurobi's model building APIs
gurobipy, C, C++, C#, Java, Matlab, R



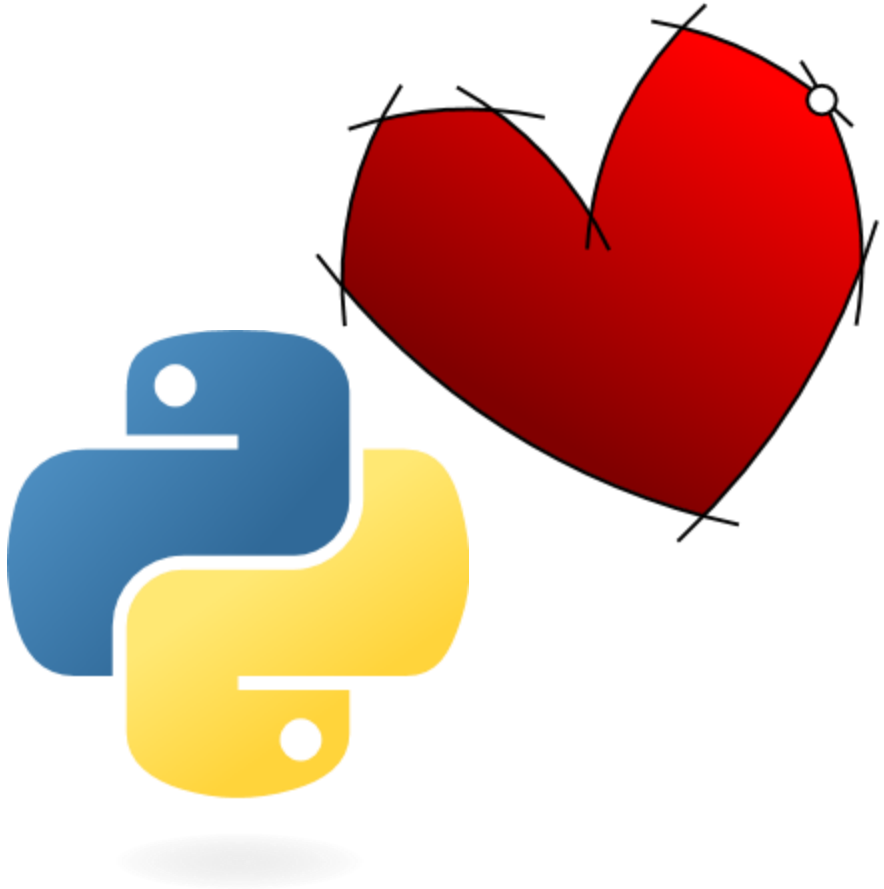
Gurobi has a heart for Python

- Gurobi's Python API "gurobipy" was part of the product from the start
- Built for fast and memory efficient model-building in Python
- Most feature-rich API to use the Gurobi Optimizer
- We continue to modernize and improve with every release!

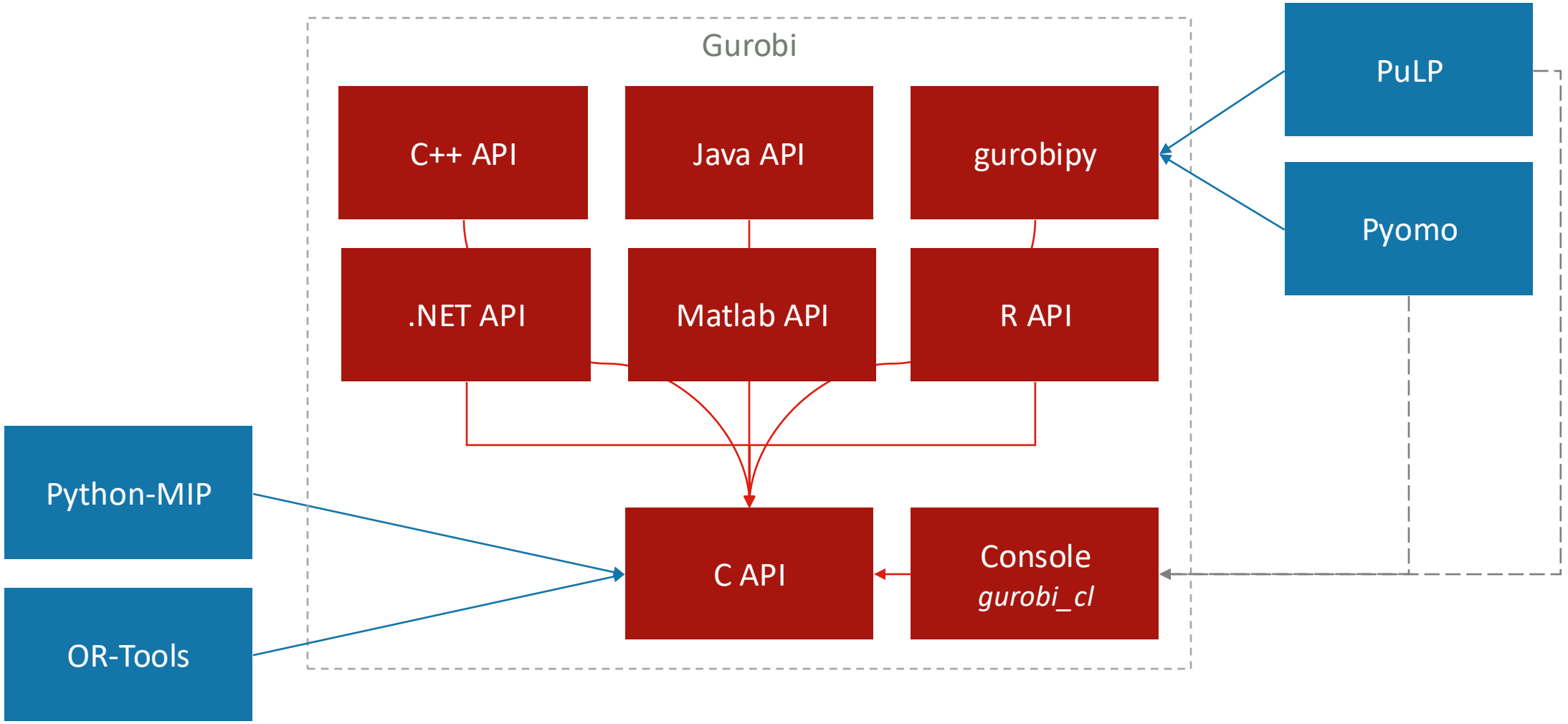
pip install gurobipy

conda install –c gurobi gurobi*

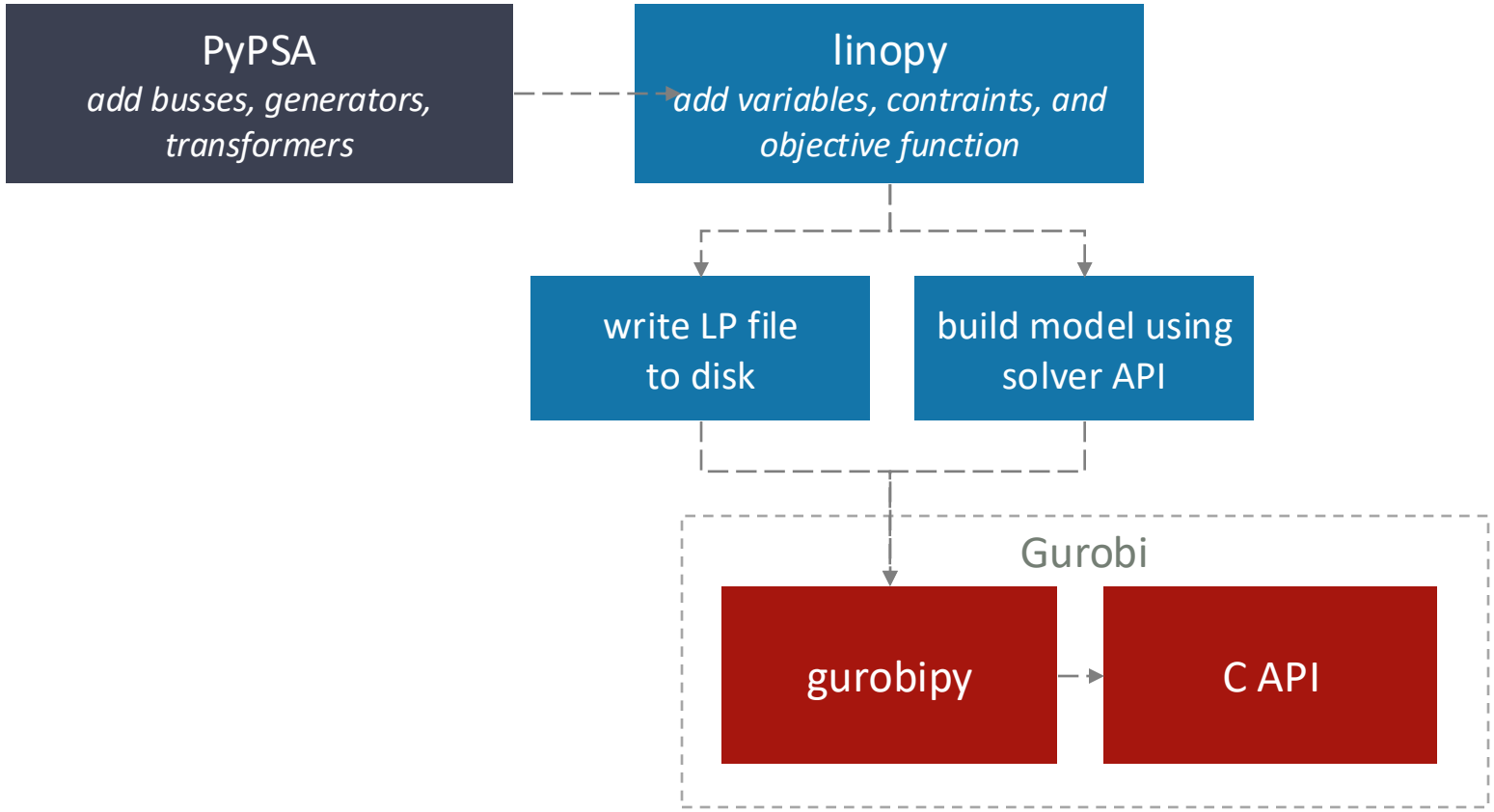
*conda install contains gurobi_cl



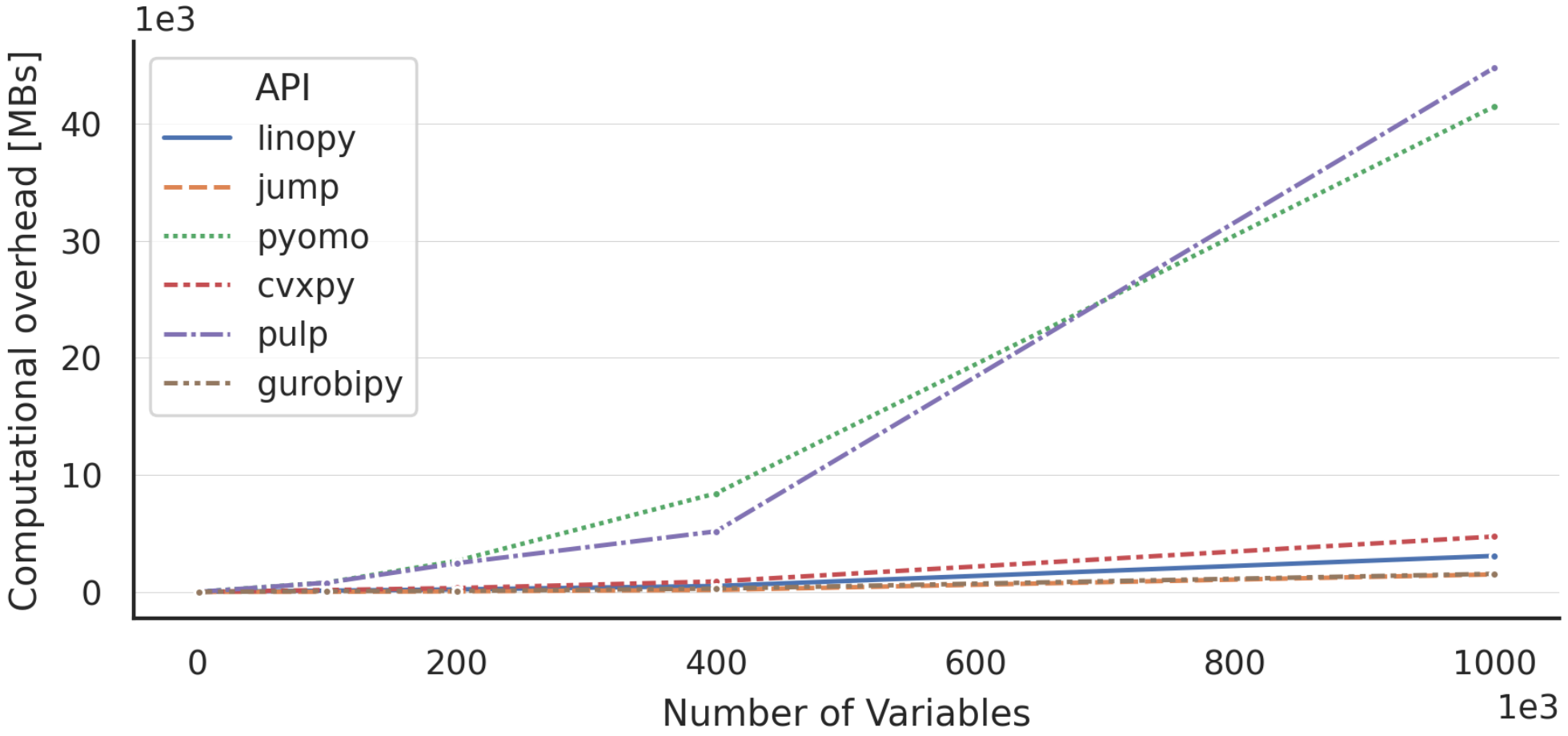
Solvers, APIs and frameworks



Connecting PyPSA to Gurobi

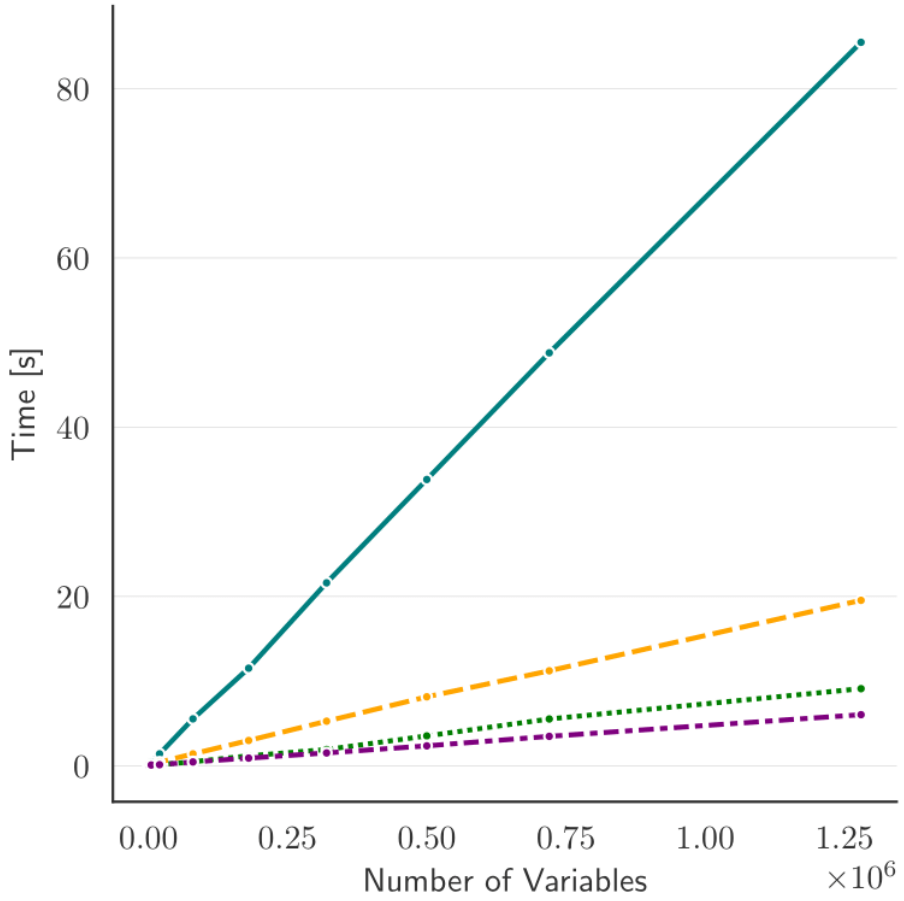


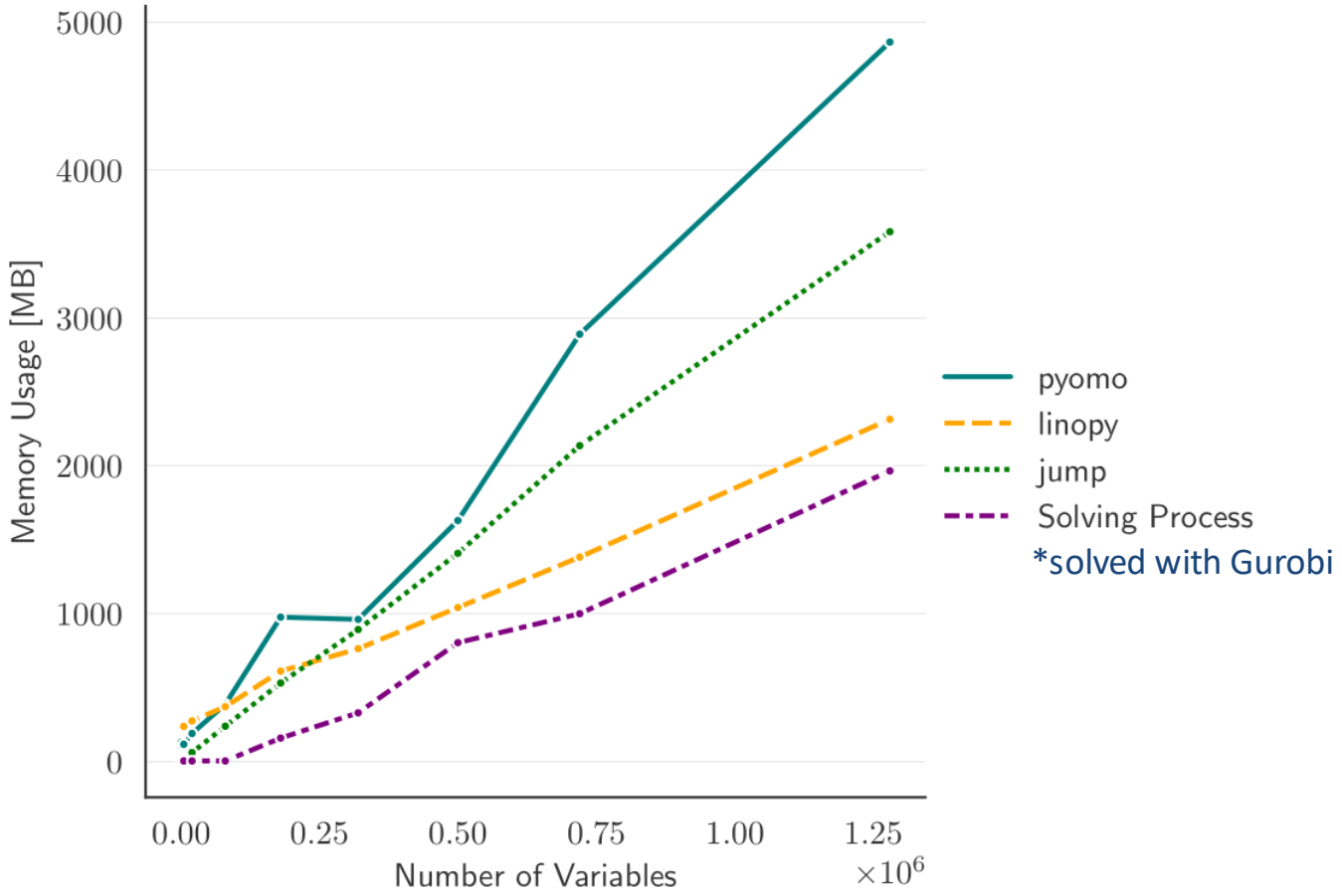
Model building memory efficiency



https://github.com/pypsa/linopy

Model building speed





https://linopy.readthedocs.io/en/latest/benchmark.html

With or without a modeling framework?

	Modeling framework	Gurobi directly	However…
Support	Depends on community	Commercial support	We do our best and can often help you
Learning curve	Use what you already know	Gurobi-specific API	Very similar to open-source frameworks
Performance	Overhead (memory and time)	Optimal	
Benchmarking	Easy; just one line of code	Need to re-implement models	Benchmark via MPS exports first
Vendor lock-in	With the modeling framework	With Gurobi as a solver	Centralize your optimization logic
Feature set	Limited to what is exposed by framework	Full access to all Gurobi features	

import gurobipy as gp

Gurobi's Python API

Four major approaches

1) Element-by-element

```
model.addConstrs(
   output.sum(‘*‘, period)
   >= demand[period]
   for period in range(nperiods))
)
```

2) Matrix-friendly API

```
m.addConstr(A @ x <= rhs)
```

3) Pandas integration

```
demand = gppd.add_constrs(
   m,
   df[‘output'].groupby(‘type').sum(),
   GRB.GREATER_EQUAL,
   demand[”Period"]
)
```

4) Machine learning integration

```
lin_reg = make_pipeline(
  feat_transform, LinearRegression()) lin_reg.fit(X_train,
y_train)
…
pred_constr = add_predictor_constr(
  m, lin_reg, features, result)
```


Efficient data structures in gurobipy

Tupledicts enable efficient variable and coefficient selection:

```
d = gp.tupledict([((1,2), 'onetwo'), ((1,3), 'onethree'), ((2,3), 'twothree')])
print(d.select(1, '*')) # prints ['onetwo', 'onethree']


x = m.addVars([(1,2), (1,3), (2,3)])
expr = x.sum()      # LinExpr: x[1,2] + x[1,3] + x[2,3]
expr = x.sum('*', 3) # LinExpr: x[1,3] + x[2,3]


coeff = dict([((1,2), 2.0), ((1,3), 2.1), ((2,3), 3.3)])
expr = x.prod(coeff) # LinExpr: 2.0 x[1,2] + 2.1 x[1,3] + 3.3 x[2,3]
expr = x.prod(coeff, '*', 3) # LinExpr: 2.1 x[1,3] + 3.3 x[2,3]
```


Efficient data structures in gurobipy

Multidict splits a single dictionary into multiple dictionaries.

```
keys, dict1, dict2 = gp.multidict( {
  'key1': [1, 2],
  'key2': [1, 3],
  'key3': [1, 4] } )
```

Quicksum is a version of the Python sum function that is much more efficient for building large Gurobi expressions.

```
expr = gp.quicksum([2*x, 3*y+1, 4*z*z])
expr = gp.quicksum(model.getVars())
```


Term-based vs. Matrix-based model building

GUROBI OPTIMIZATION

```
# Term Based
m=gp.Model()
x=m.addVar(ub=1.0)
y=m.addVar(ub=1.0)
z=m.addVar(ub=1.0)
m.setObjective(x*x + 2*y*y + 3*z*z)
m.addConstr(x + 2*y + 3*z >= 4)
m.addConstr(x + y >= 1)
```

```
# Matrix Based
m=gp.Model()
Q=np.diag([1,2,3])
A=np.array([[1,2,3],[1,1,0]])
b=np.array([4,1])
x = m.addMVar(3, ub=1.0)
m.setObjective(x @ Q @ x)
m.addConstr(A@x >= b)
```


gurobipy's matrix-friendly modeling API

- Enables natural model building using matrix-based expressions
- Complements existing capabilities for term-based modeling
- Concepts and semantics lean on NumPy

```
expr1 = A @ x
expr2 = A @ x + B @ y + z
expr3 = x @ A @ x + y @ B @ y
```

NumPy

Powerful N-dimensional arrays

Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.

gurobipy-pandas

gurobipy-pandas Convenient wrapper library to connect pandas with gurobipy.

pandas

- Efficiently build mathematical optimization models from DataFrames and Series objects.
- Built for experienced pandas users who are familiar with methods to transform, group, and aggregate data stored in DataFrames.
- Some familiarity with optimization modeling is required but does not require deep experience with gurobipy.

```
pip install gurobipy-pandas
```


gurobipy-pandas

GUROBI OPTIMIZATION

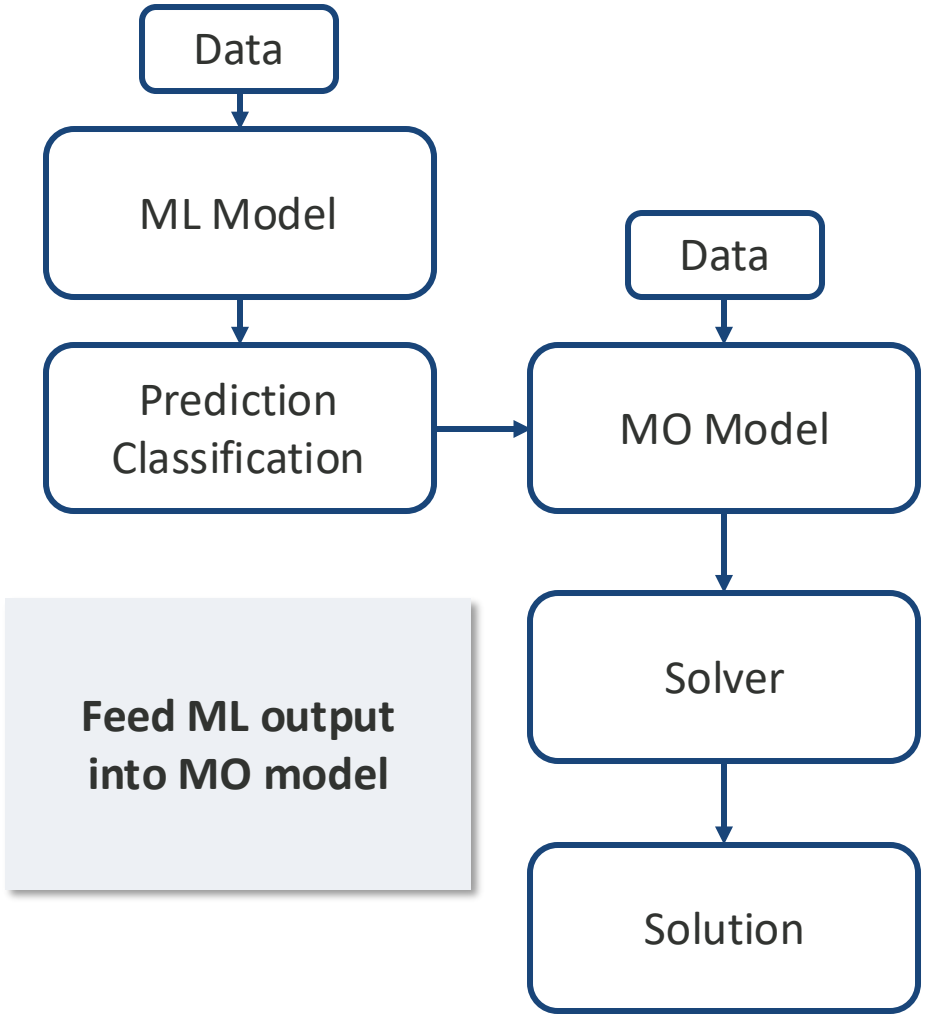
```
df.gppd.add_vars(model, name="output)
   .gppd.add_vars(
                        model, vtype=GRB.INTEGER, ub="num_available",
            name="num_active”
   )

df.gppd.add_constrs(
   model,
   "output >= min_output * num_active”)
```

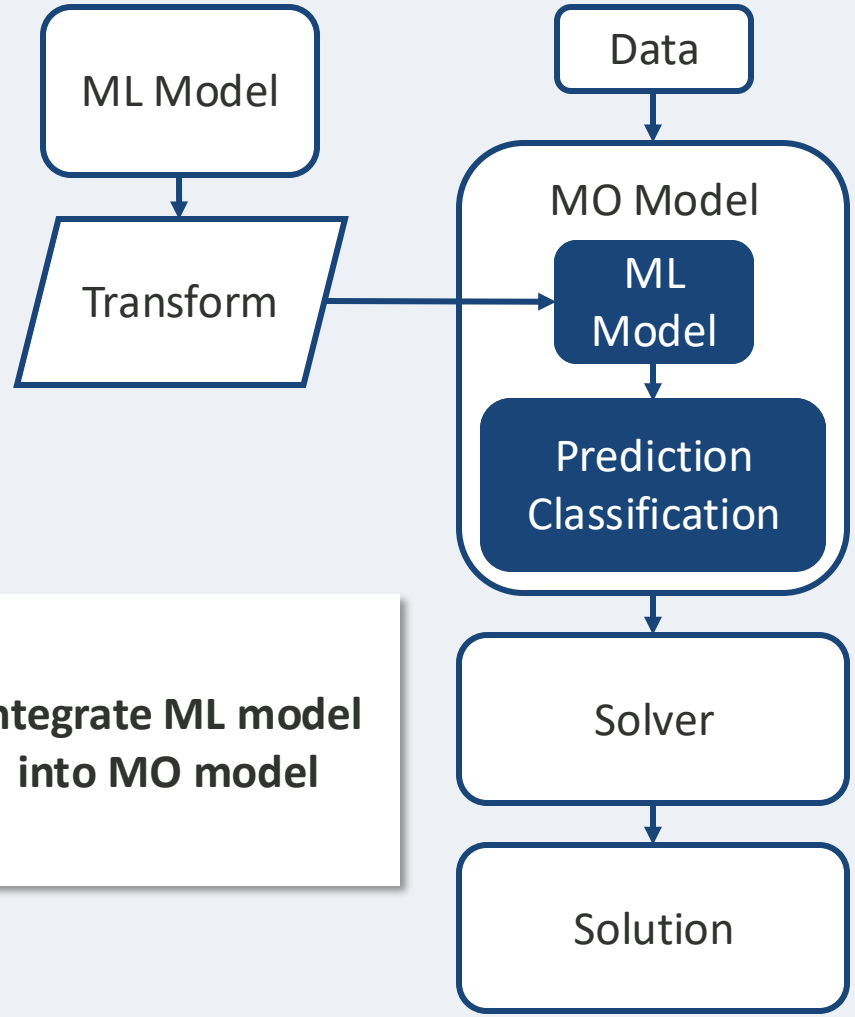

Machine Learning and Mathematical Optimization

Joining forces

Working hand in hand



Feed ML output into MO model



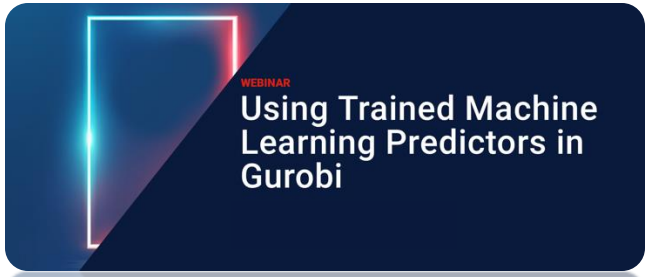
Integrate ML model into MO model

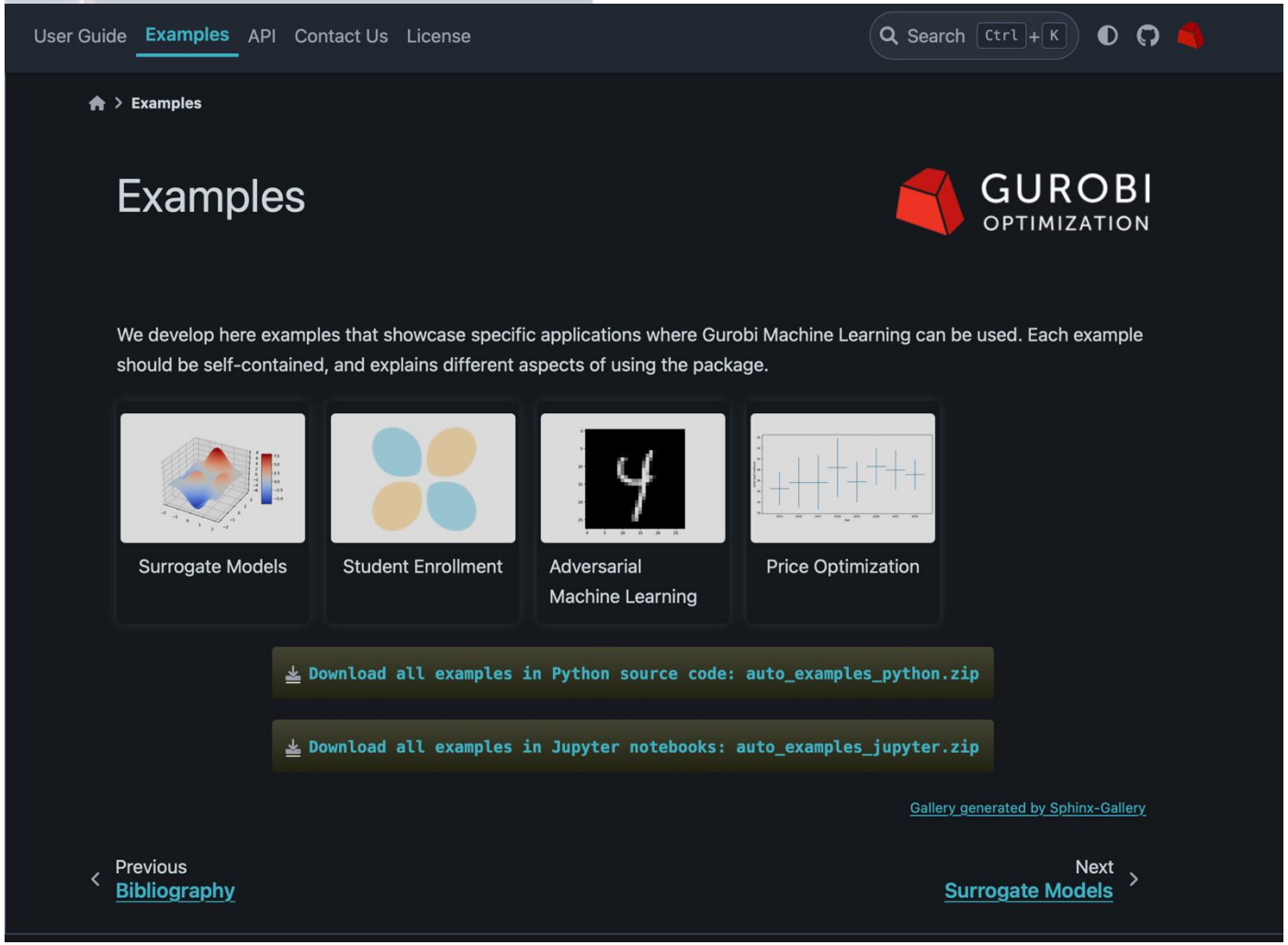
GUROBI OPTIMIZATION

Gurobi Machine Learning

Incorporate trained regression models into optimization models

- Input and output of the ML model are decision variables of the optimization
- Feasible solutions of the optimization obey the regressor's input-output relationship
- Supports scikit-learn, LightGBM, XGBoost, Keras, TensorFlow and PyTorch





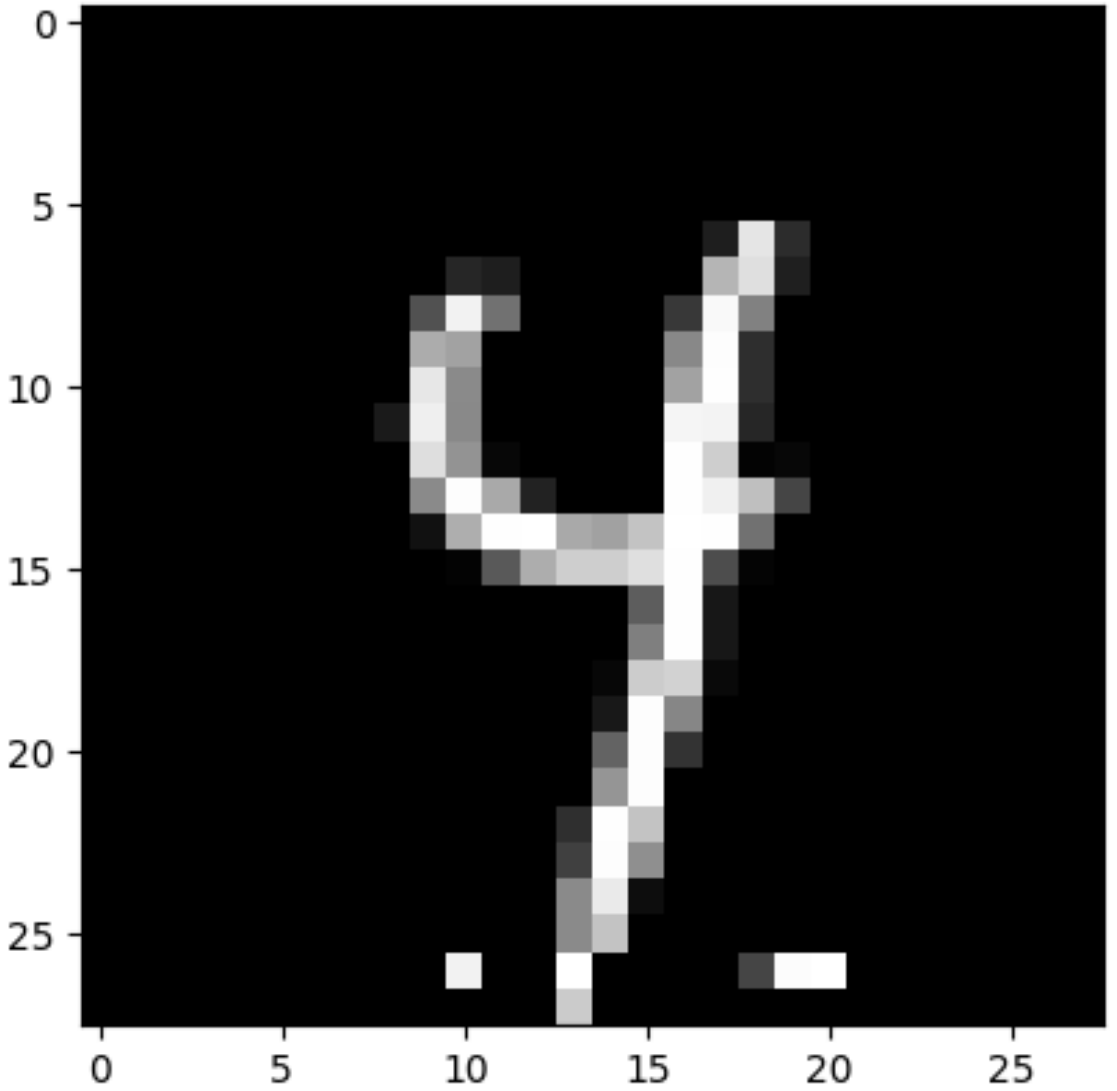
GUROBI OPTIMIZATION

Adversarial Learning

Identify the minimum change required to your input data to break your classification model

- Scikit-learn's MLPRegressor classifies handwritten digits
- An optimal solution manipulates a given input as little as possible while causing misclassification
- The letter on the right is classified as 9

Gurobi - Adversarial Machine Learning



Gurobi Machine Learning

GUROBI OPTIMIZATION

Open-source package to formulate trained ML predictors with Gurobi

```
1 from gurobi_ml import add_predictor_constr
2
3 add_predictor_constr(gp_model, ml_predictor, input_vars, output_vars)
```

A gurobipy speciality: Session boundaries

Session boundaries

An environment that is not disposed, keeps a session active (also blocks a license token).
Especially important in interactive environments.

```
with gp.Env() as env:
  with gp.Model(env=env) as model:
    # construct, solve, and post-process `model`
    # ...
```

Alternative

```
…
model.dispose()
env.dispose()
# disposeDefaultEnv() # if environment was not created explicitly before
```

For small models, re-using one environment for a sequence of models is way faster.
You can create and manage a **gurobipy** environment with many other model building frameworks.

Questions?

Jaromił Najman

Senior Developer

