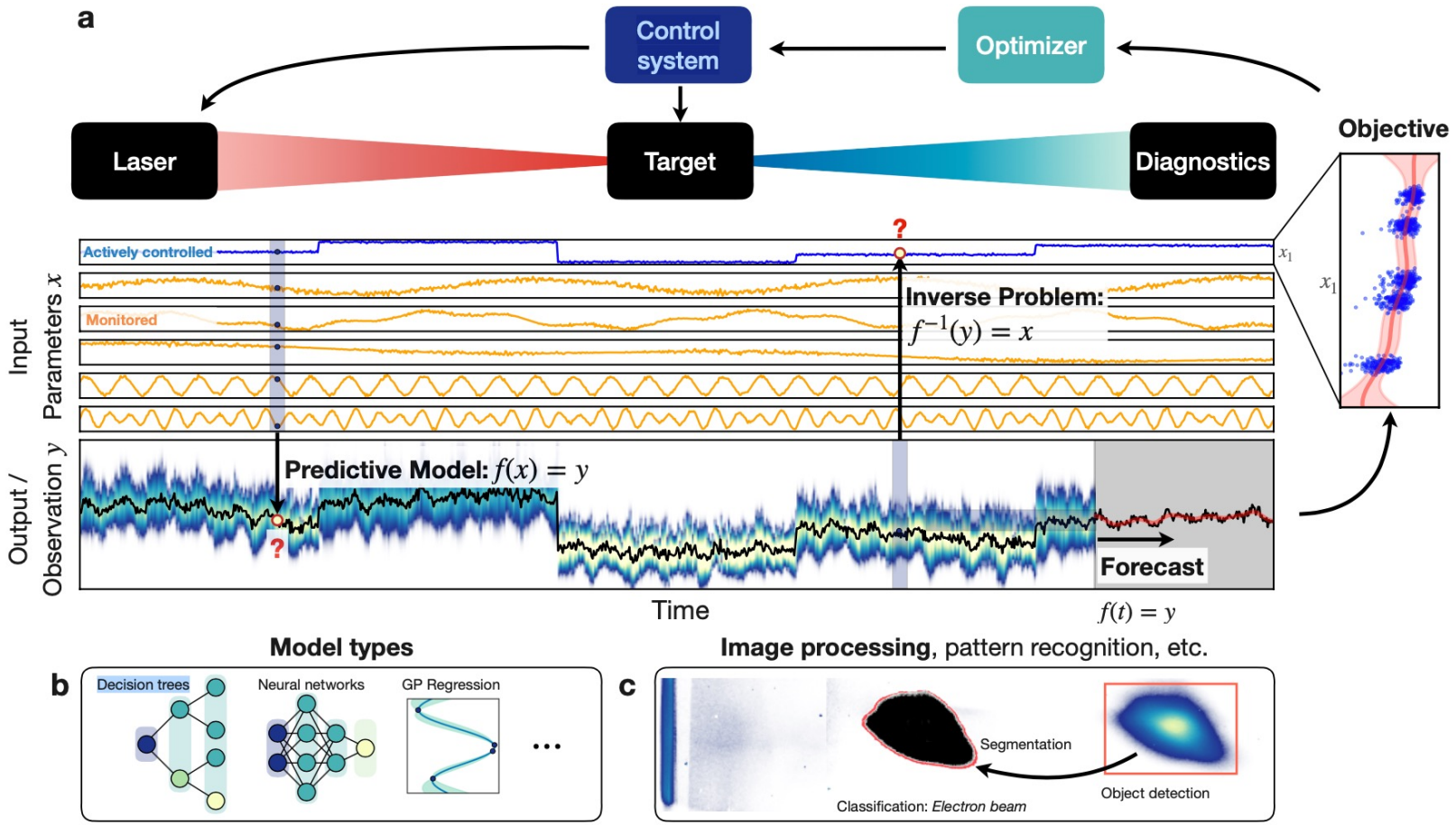
Optimisation and surrogate models for PALLAS project

V. Kubytskyi on behalf of PALLAS project

ML in LPA research

- ML in modelling and experiment in accelerator physics.
- B.O. , NN, feedbacks, fast simulations.
- Data driven studies



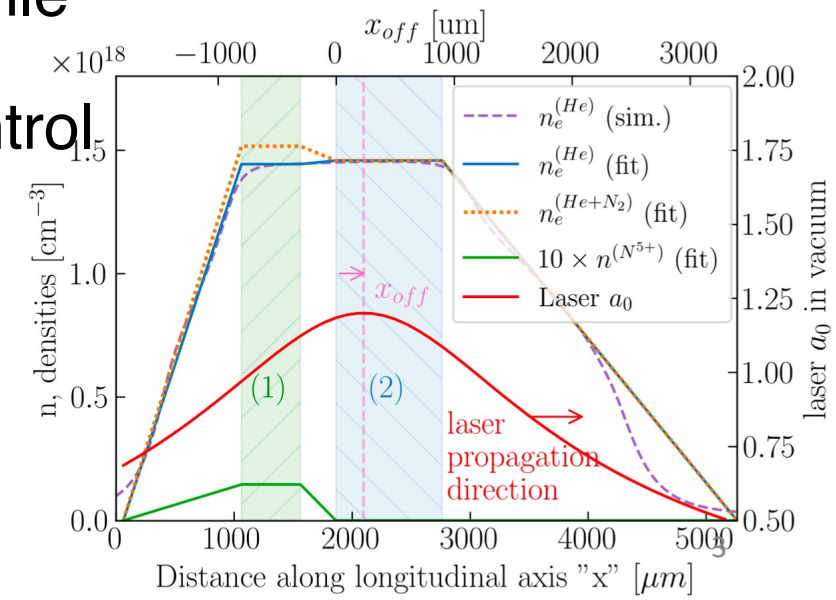
A Dopp et al., Data-driven Science and Machine Learning Methods in Laser-Plasma Physics https://arxiv.org/pdf/2212.00026.pdf

Pallas experiment at IJCLab

Parameters	phase 1	phase 2	phase 3	unit
energy	150	**200**	200	MeV
charge	15-30	**30**	**>30**	pC
frep	10	10	10	Hz
energy spread	<5%	**< 3%**	< 2%	rms
$\varepsilon_{n,rms}$	1	**<1**	**<1**	μm
stability	5%	**3%**	**<1%**	-
reproductibility	5%	**3%**	<3%	-

Build a **laser-plasma injector** (**LPI**) prototype with **reliability** and **control** comparable to conventional **RF accelerator** standards.

- plasma cell parameters, tailored plasma density profile
- laser transport, focalization and advanced laser control
- e- beam transport, characterization

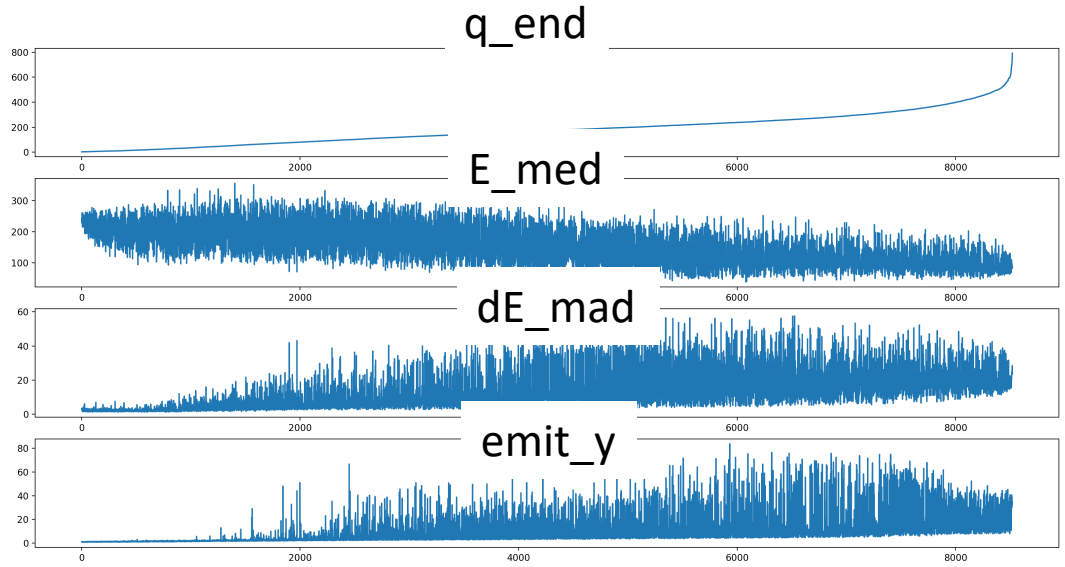


Dataset

- “free” . contains random scans. Size ~15000.

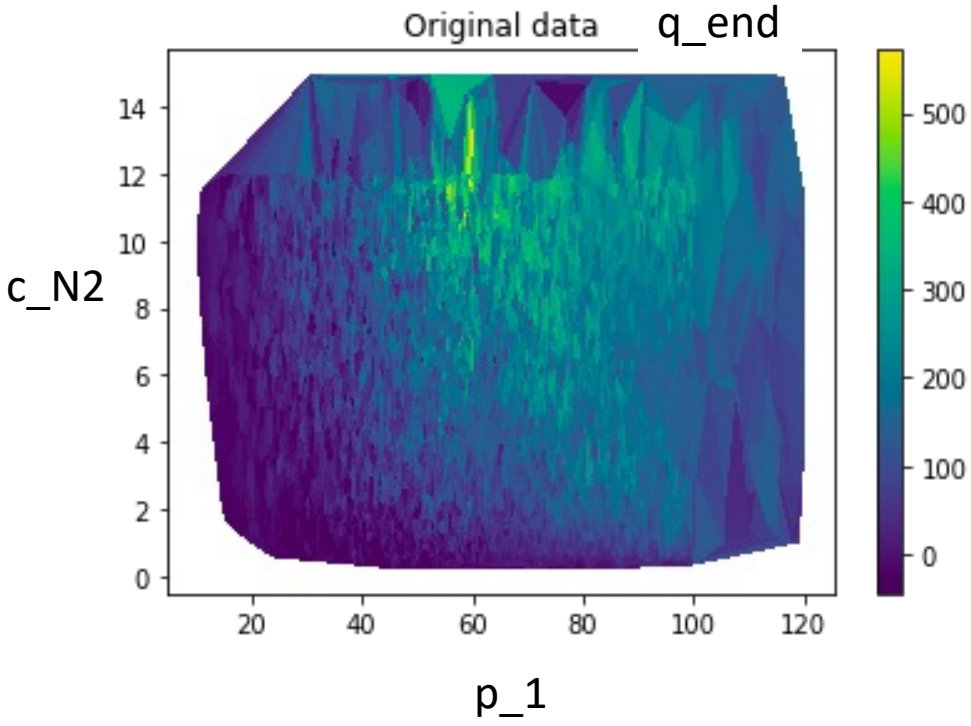
- Inputs: x_of, a_0,c_N2, p_1

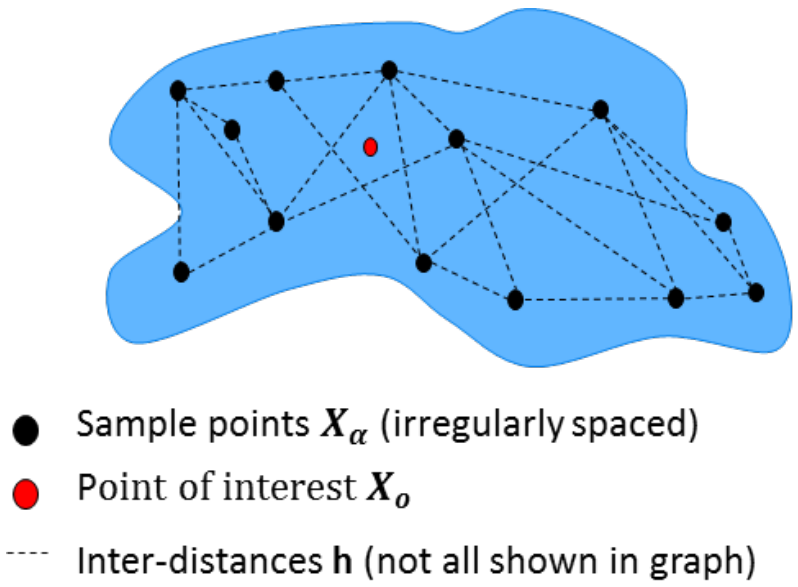
- Outputs: E_med , dE_mad , q_end , emit_y .
  Can contain more beam parameters, objective functions, etc.



	x_of	p_1	c_N2	a_0	q_end	E_mean	E_med	E_std	E_peak	emit_y	emit_z	dQdE_max	obj1
87	1048.124474	37.948624	0.120637	1.070767	8.339986	174.017081	176.025811	6.348132	177.006592	0.865049	0.231377	0.778338	0.341051
329	1063.754900	48.468193	0.071831	1.036420	7.540230	209.237908	210.579185	4.796401	211.054592	0.951493	0.246727	0.619934	0.302193
160	1104.597541	47.714470	0.073125	1.035019	5.084618	210.132015	211.558100	3.473369	211.054592	0.927990	0.219252	0.473531	0.298801
52	587.024874	30.974860	0.082971	1.159662	9.312980	161.606688	163.060832	6.002437	164.846592	0.916083	0.273261	0.873819	0.278469
433	918.899055	47.102972	0.046897	1.083746	9.289970	214.683598	215.436486	3.824135	215.918592	1.112371	0.288656	0.752876	0.247436

Problem





E_med , dE_mad , q_end , emit_y = **F**(x_of, a_0,c_N2, p_1) **Possible**

x_of, a_0,c_N2, p_1= $\mathbf{F}^{-1}$ (E_med , dE_mad , q_end , emit_y) **Difficult**

Surrogate model

Main idea: (re)use knowledge of already computed configurations (scans, BO, random search) and obtained beam parameters to construct **surrogate model of the accelerator**.

Quickly probe different configurations, perform fast **optimization** in global parameters space, estimate uncertainty. Better understand our data, relations between features.

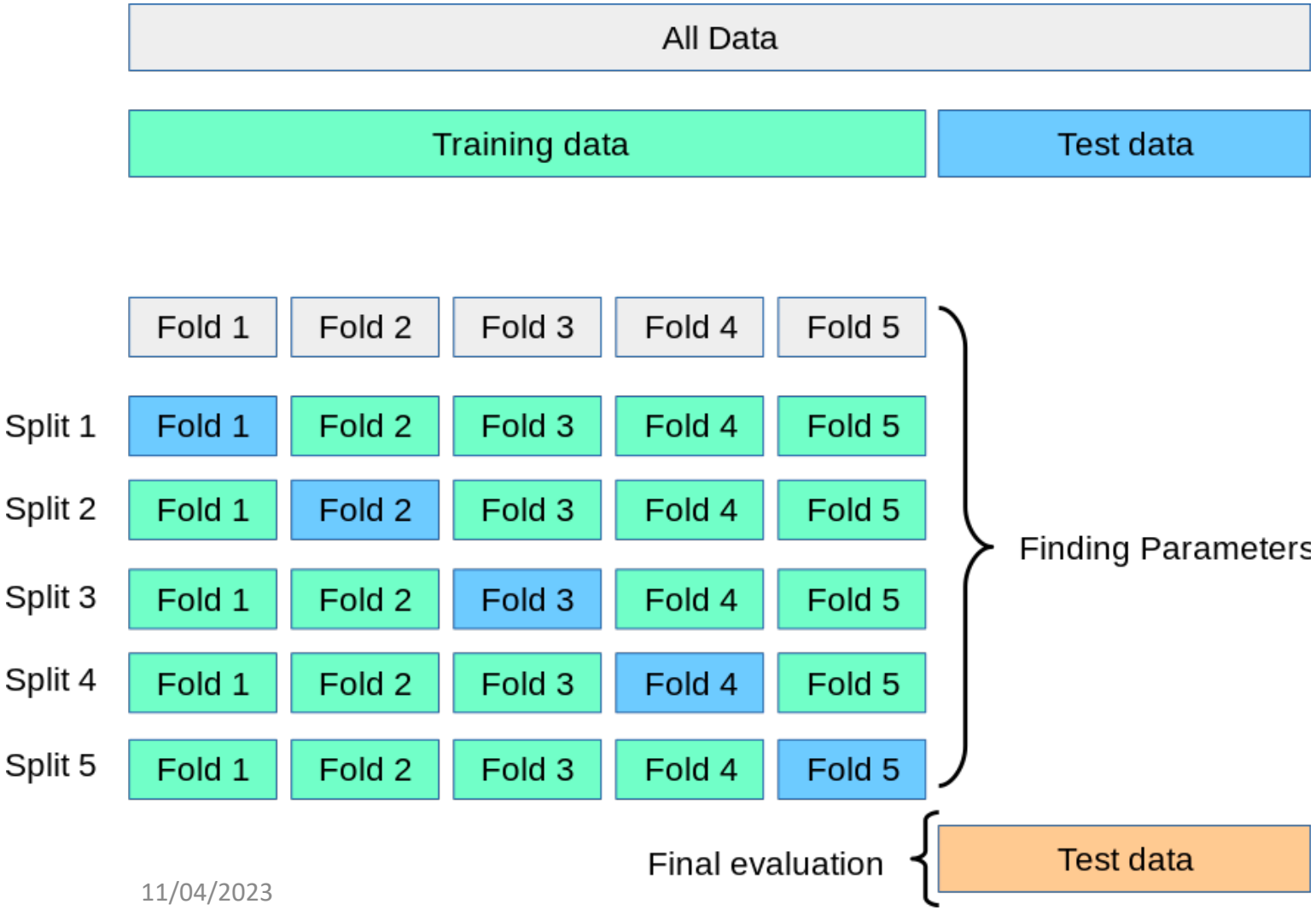
New configuration can be estimated from the surrogate model and validated with SMILEI. New refined data is then inserted in the dataset to continuously improve surrogate model.

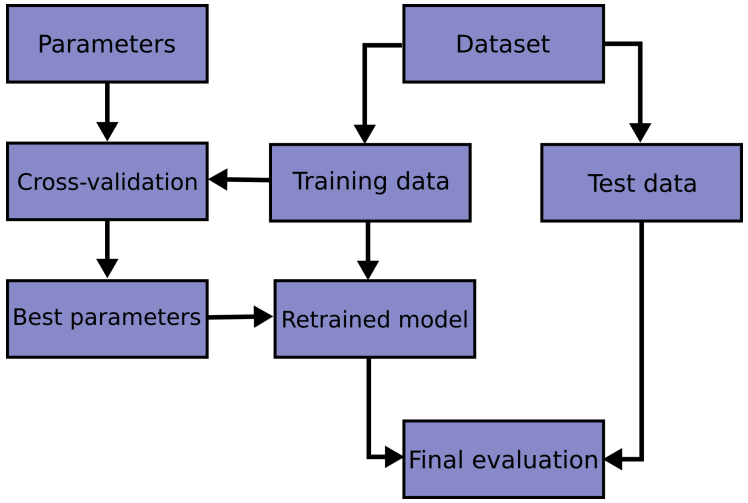
Tested ML methods

- Neural Networks
- XGBoost
- Gaussian Processes Gpy

Pros and Cons methods.

K-Folds cross-validation

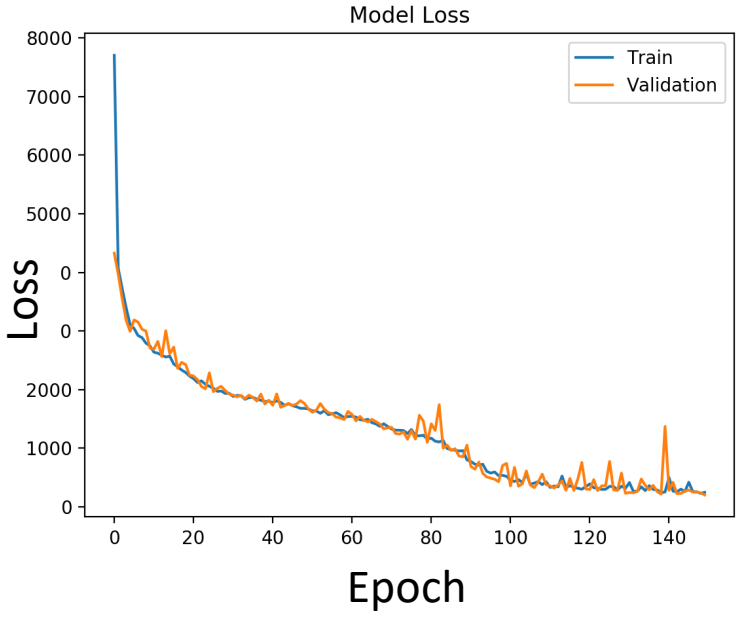


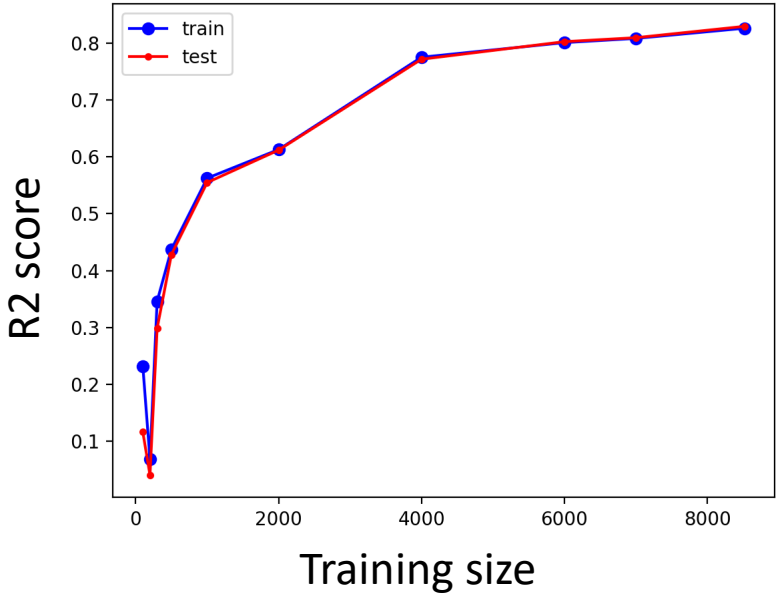


Neural Network

```
model = Sequential()
model.add(Dense(64, input_shape=(4,), activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(4, activation='linear'))
model.compile(optimizer='adam', loss='mean_squared_error')
```

- Training , overfit, Kfolds, R2score





$$\mathrm{MSE}(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

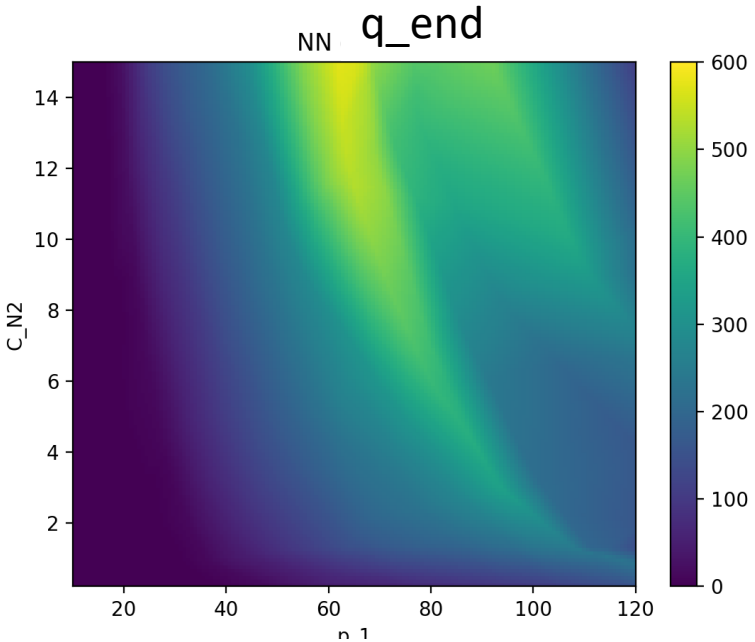
$$\mathrm{R}^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

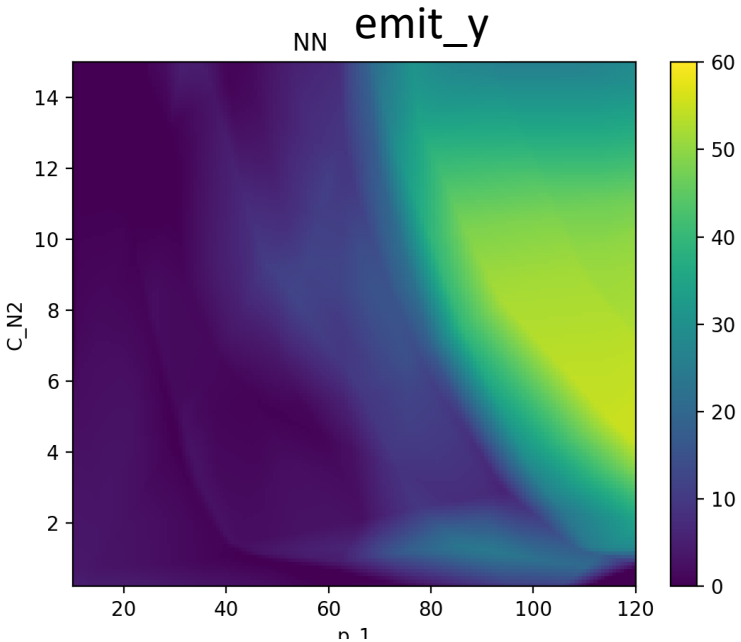
MSE_10KFolds = [593.0, 118.9, 217.9, 144.9, 86.1, 58.7, 124.0, 54.9, 62.7, 83.2]

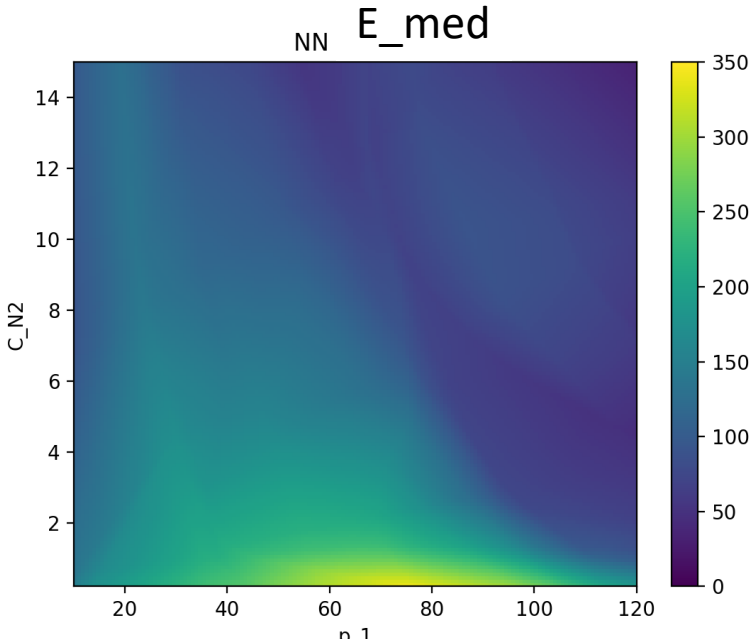
Model:
E_med , dE_mad , q_end , emit_y = **F**(x_of, a_0,c_N2, p_1)

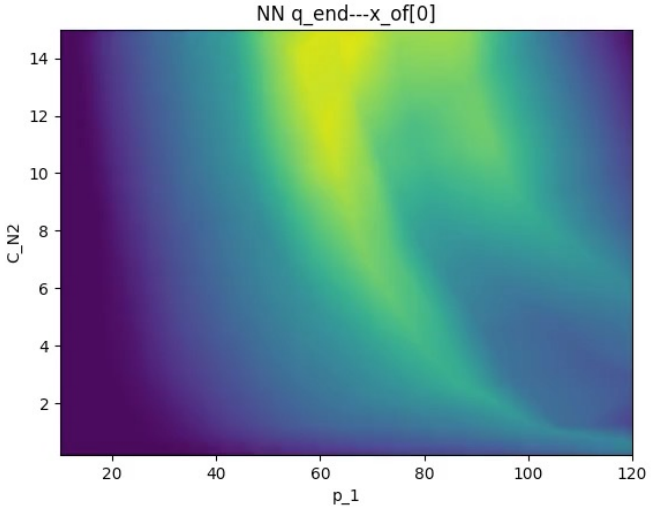
R2 score:
Test = 0.914
Train = 0.917

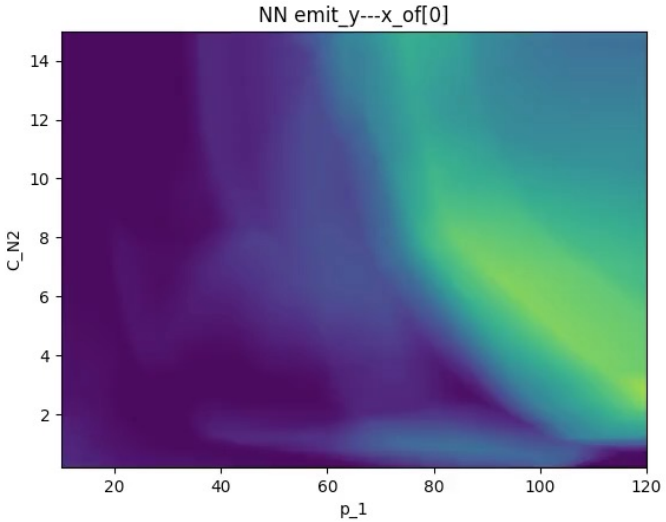
Prediction maps with NN

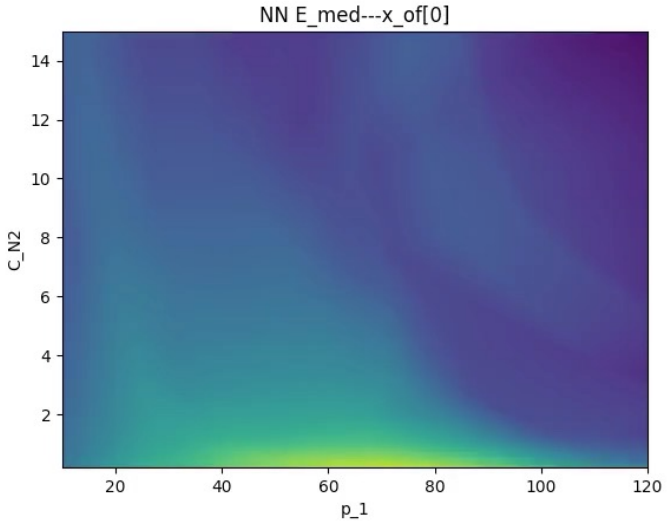




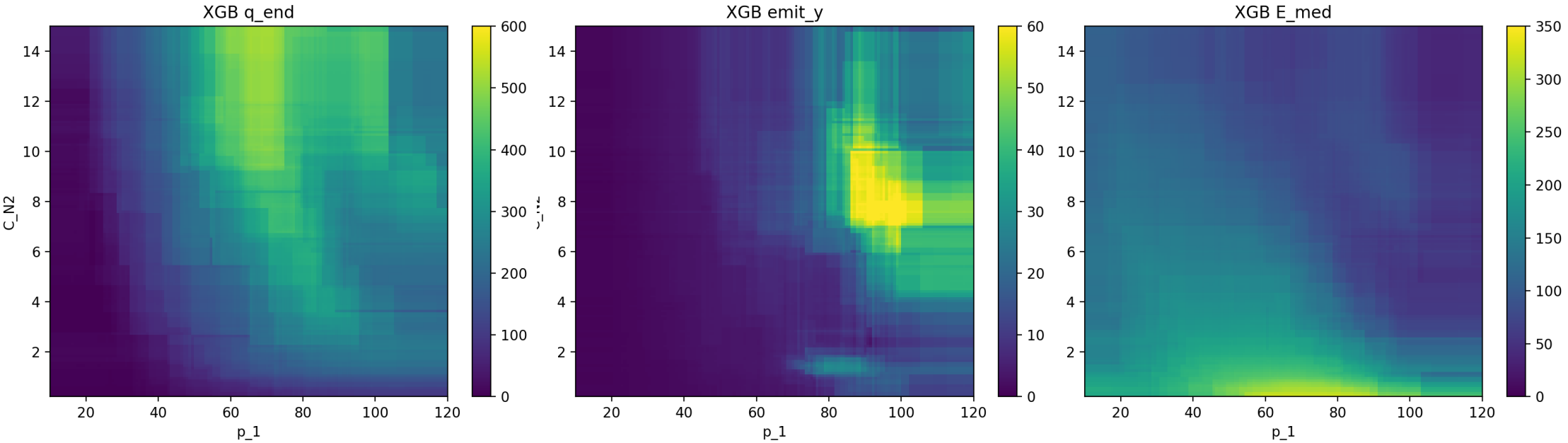








XGBoost

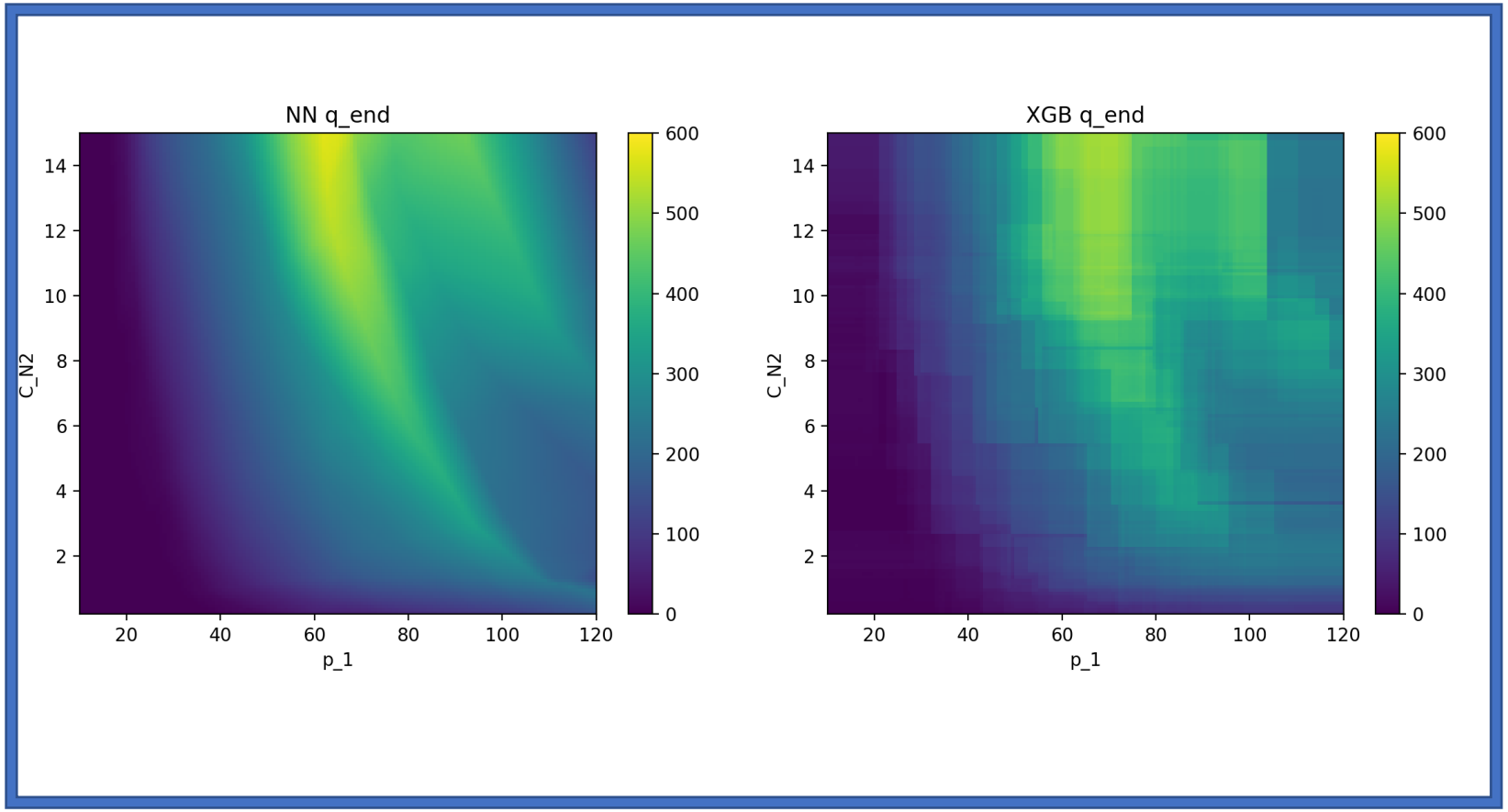


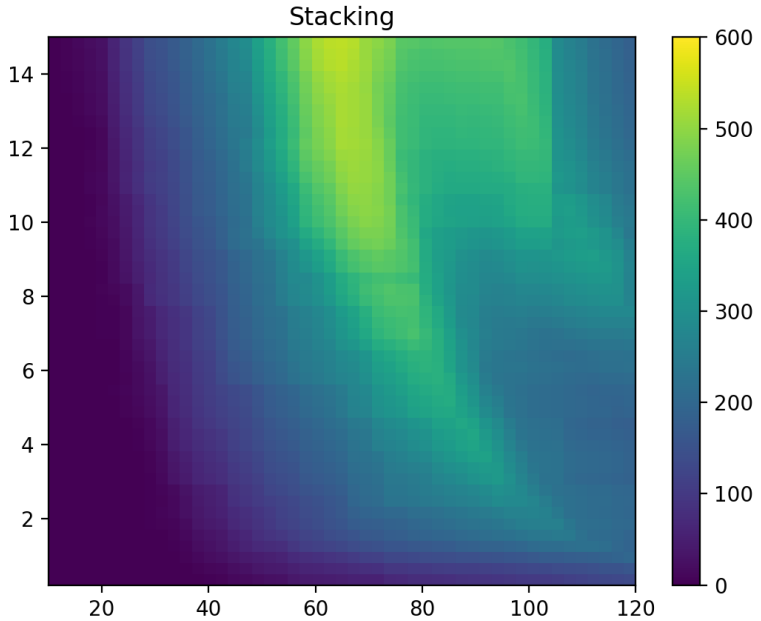
Feature importance
{‘x_of': 706,
‘a_0': 2510,
‘c_N2': 5255,
‘p_1': 4744}

Mean Squared Error:
[141.3, 119.1, 95.4, 124.4, 106.8, 120.2, 109.4, 110.6, 105.3, 109.2]

R2 score
Test 0.938
Train 0.993

Combine in best way NN with Xgboost. Stacking



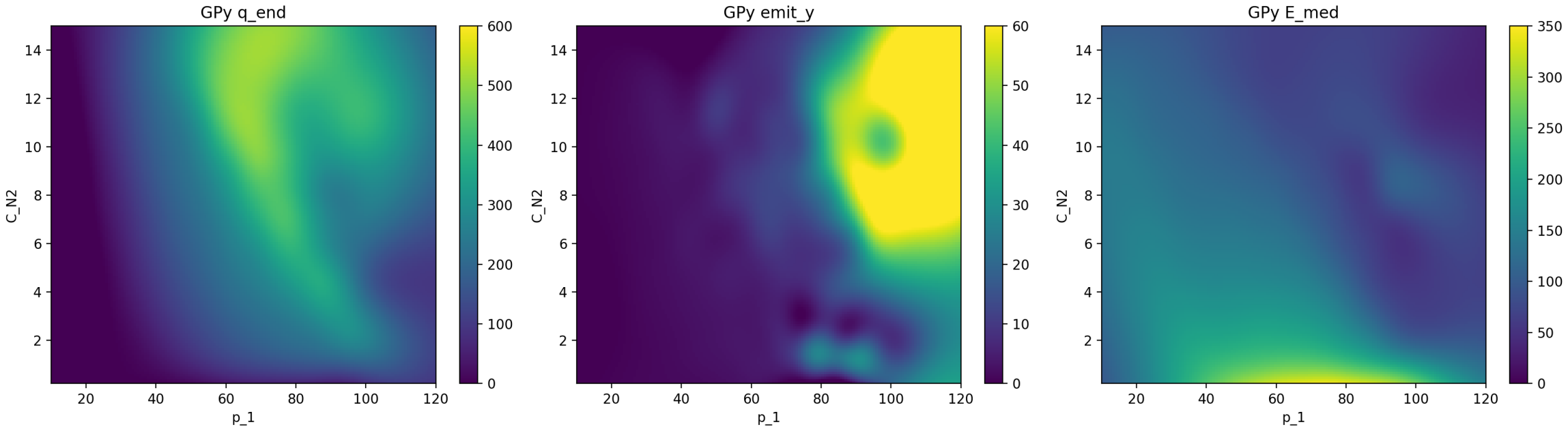


R2 score

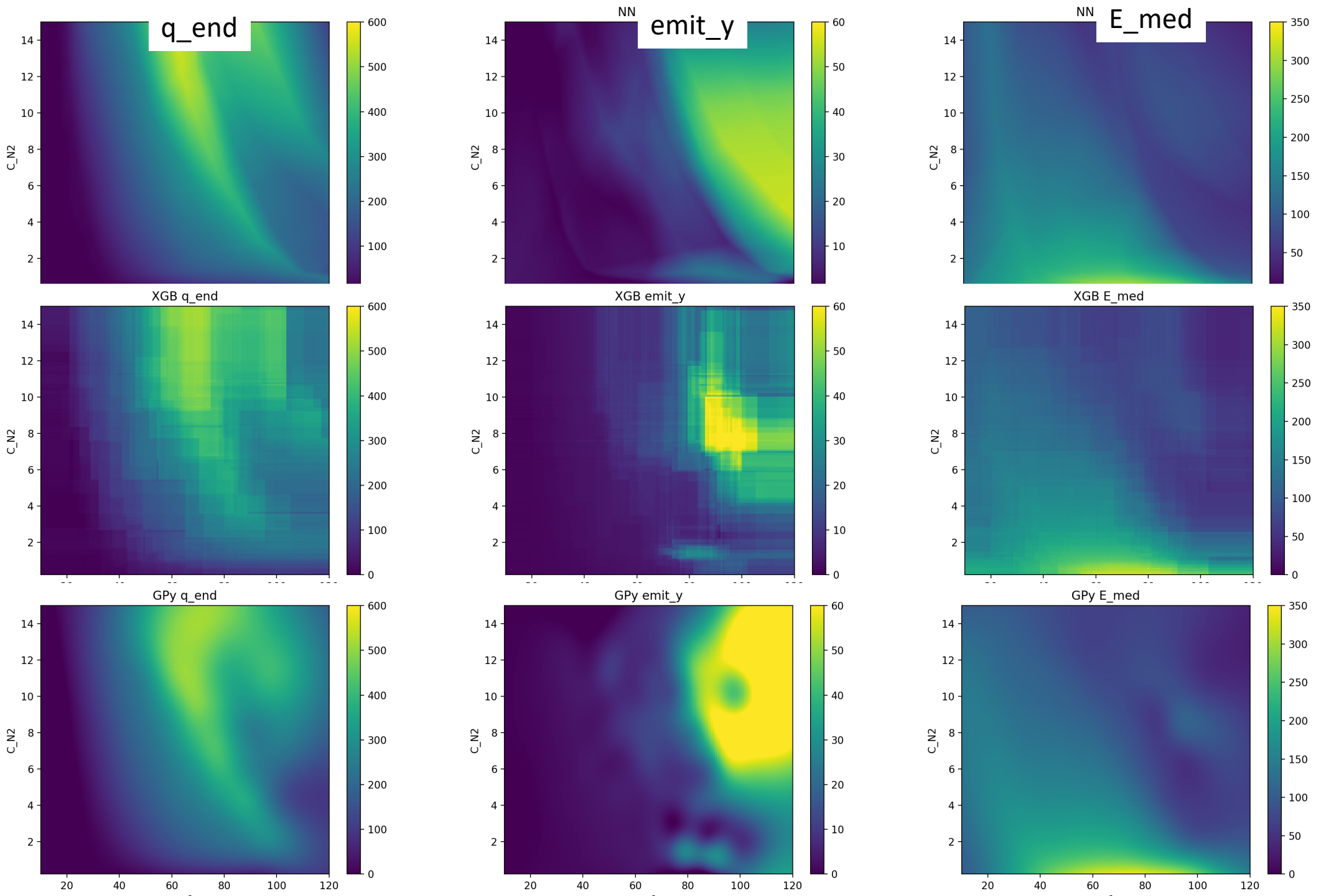
0.986

GPy

- Probabilistic approach, noise resistant, choice kernel.



R2 score
Test 0.93
Train 1.0


q_end
emit_y
E_med
NN
XGB q_end
XGB emit_y
XGB E_med
GPy q_end
GPy emit_y
GPy E_med
C_N2

NN

XGB

GPy

Summary table

	MODEL	Input	output	R2train	R2test	dataset	File, speed
1	NN	4	4	0.917	0.914	full	NN_with_KFold
2	XGBoost	4	4	0.99	0.93	full	XGBoost_with_KFold
3	GPy	4	4	1	0.93	50%	GPy
4	stacking	4	4		0.98	full	stacking_XGBoost_and_NN_with_KFold
5	NN inverse	4	4	0.52	0.5		
6	XGBoost inverse	4	4	0.95	0.7		

Optimization search with trained models

Objective function : $\sqrt{Q}$ /(E_med ΔE)

optimizer: from scipy.optimize import differential_evolution

Bounds: "x_of" (0, 2937) "a_0" (1, 1.845) "c_N2" (0.21, 14.6) "p_1" (10.58, 119.6)

- Neural Networks -253493.9 "x_of" 1620.51 "a_0" 1.14 "c_N2" 4.31 "p_1" 58.37

[[2.3106668e+02 1.9192696e-05 2.3670479e+01 4.5123029e+00]]

- XGBoost -4.5447083 "x_of" 1981.30 "a_0" 1.44 "c_N2" 9.45 "p_1" 36.04

[[158.82722 1.9011413 74.65189 2.5726383]]

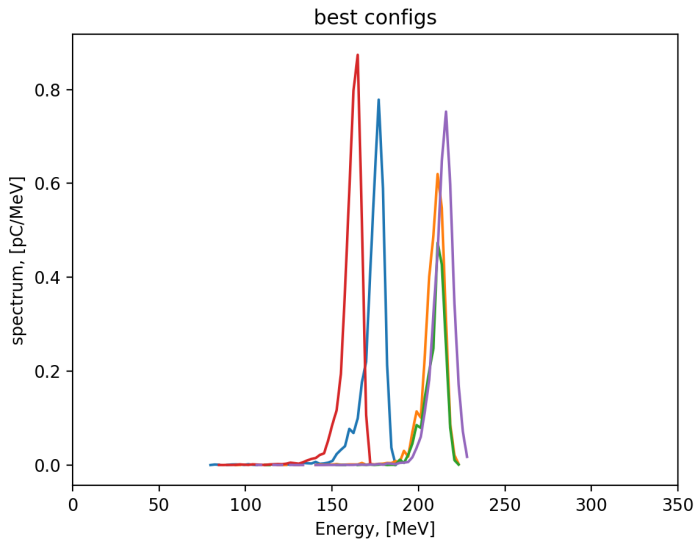
- Gpy: -28110.78158018578 "x_of" 2342.94 "a_0" 1.34 "c_N2" 9.34 "p_1" 54.20

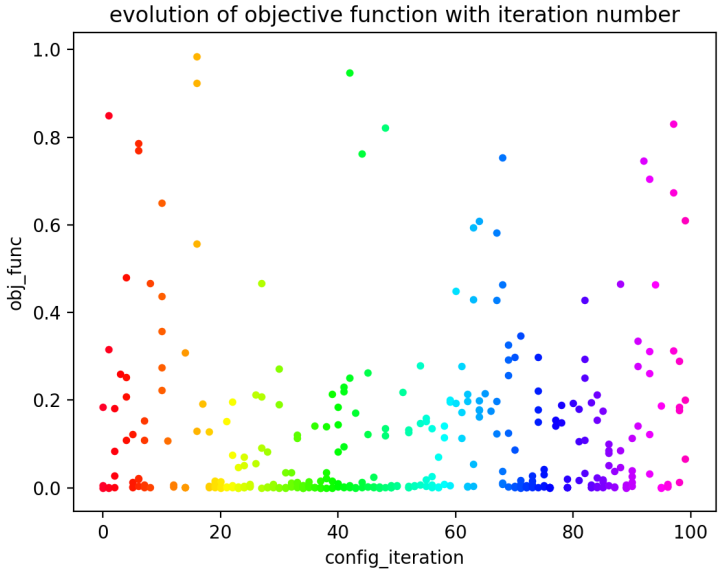
[[2.05387783e+02 2.76492111e-04 6.04103470e+01 -4.01001996e+00]]

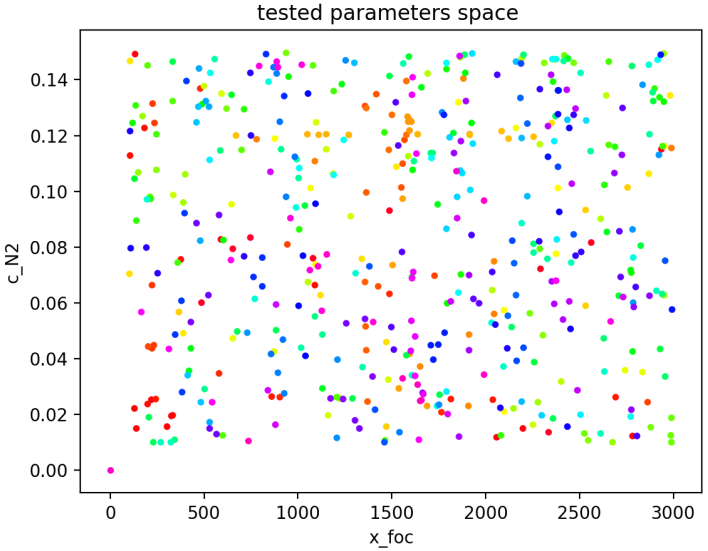
Bayesian optimization

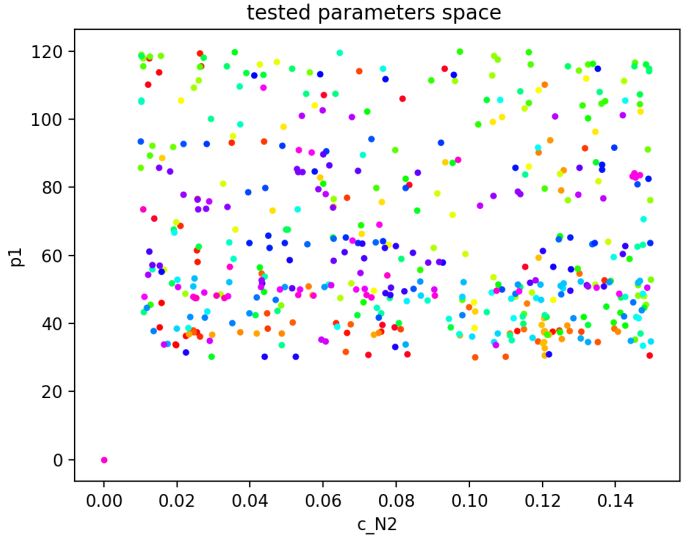
100 iterations each of 10 samples
Run at TGCC Irene

	x_of	p_1	c_N2	a_0	q_end	E_mean	E_med	E_std	E_peak	emit_y	emit_z	dQdE_max	obj1
87	1048.124474	37.948624	0.120637	1.070767	8.339986	174.017081	176.025811	6.348132	177.006592	0.865049	0.231377	0.778338	0.341051
329	1063.754900	48.468193	0.071831	1.036420	7.540230	209.237908	210.579185	4.796401	211.054592	0.951493	0.246727	0.619934	0.302193
160	1104.597541	47.714470	0.073125	1.035019	5.084618	210.132015	211.558100	3.473369	211.054592	0.927990	0.219252	0.473531	0.298801
52	587.024874	30.974860	0.082971	1.159662	9.312980	161.606688	163.060832	6.002437	164.846592	0.916083	0.273261	0.873819	0.278469
433	918.899055	47.102972	0.046897	1.083746	9.289970	214.683598	215.436486	3.824135	215.918592	1.112371	0.288656	0.752876	0.247436









Parallel bayesian search

Conclusions

- Performed in-depth study with ML in context of PALLAS, better understood our data, parameters. Reasonable size dataset.

- All techniques have shown good performances, optimization with each model.

Backup

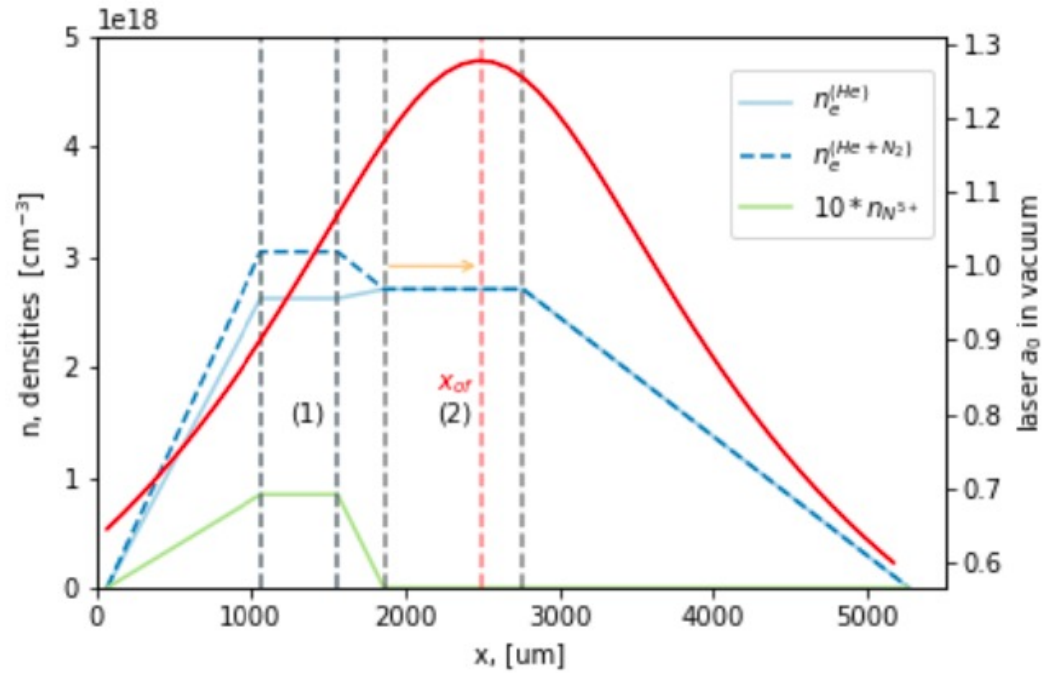
Fast PIC simulations for target configuration optimization

Numerical setup

- target configuration $x = [\, a_0,\, x_{of},\, p_1 = p_2,\, c_{N_2} \,]$
- Smilei open source PIC code with envelop approximation coupled to ionization[1], azimuthal modes[2] and low ppc :
  ~ 12-24 h.cores/mm.(*running with 240 cores on 96k computing-core HPC SKL*)
- simplified profiles from CFD simulations and confined high-Z dopant.

Two types of complementary approach:

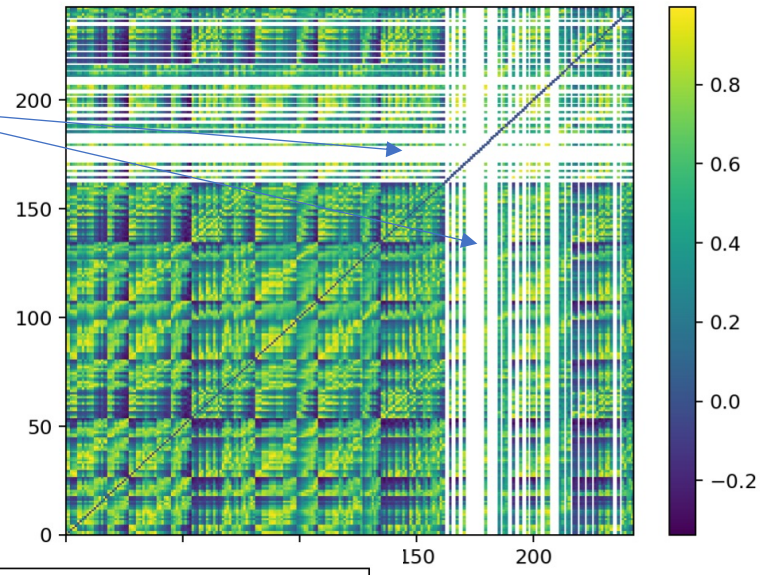
- **random scan**
- bayesian optimization



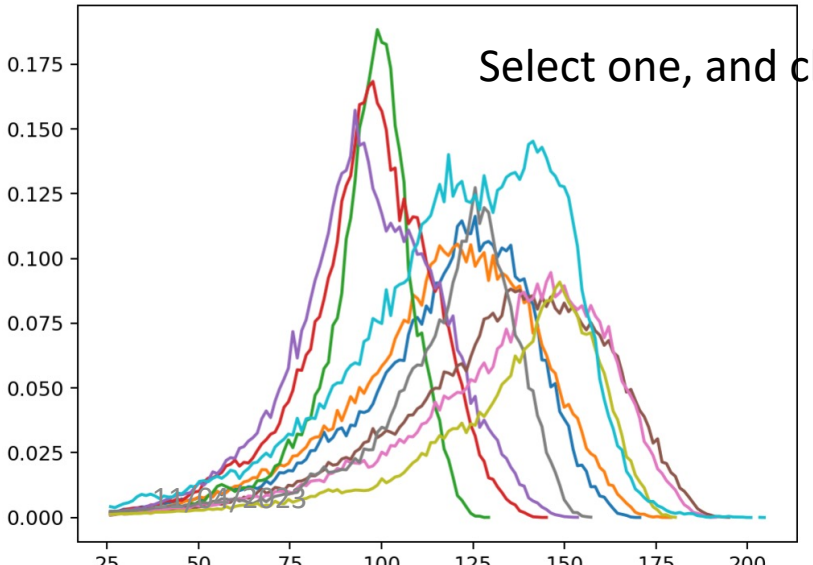
https://smileipic.github.io/Smilei/

Correlation, JSD distance (similarity between distributions) give similar results, but not simple to interpret.
Need to apply cuts, but now in correlation coefficient.

NANs



Select one, and check for the closests



K-means clustering (simple):