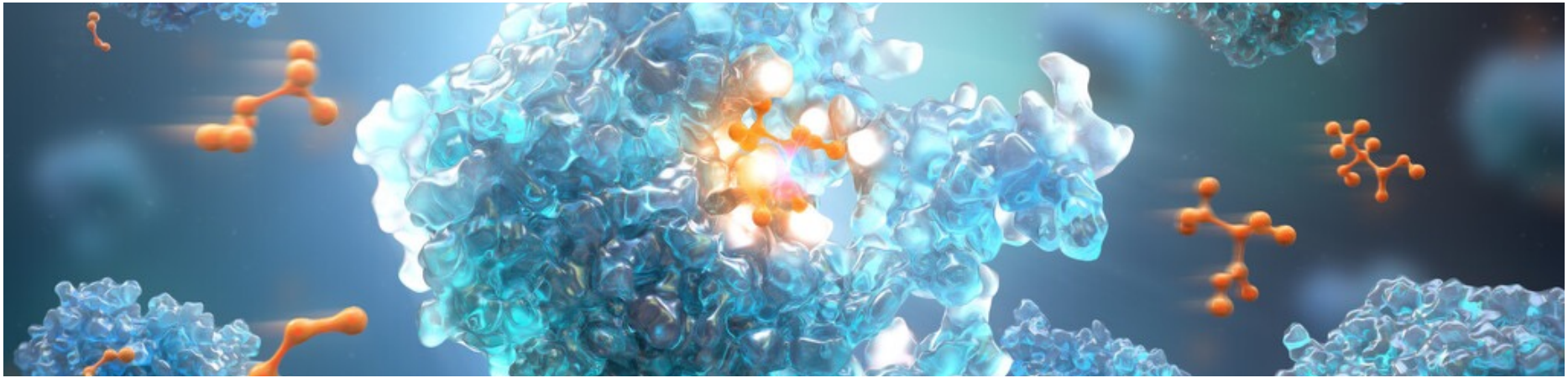
Qptuna: easy, automatic QSAR modelling

Lewis Mervin
Molecular AI, Discovery Sciences, AstraZeneca R&D, Cambridge, UK



Overview - Present

- What is Qptuna?
- Current algo’s/descriptors/features
- GUI

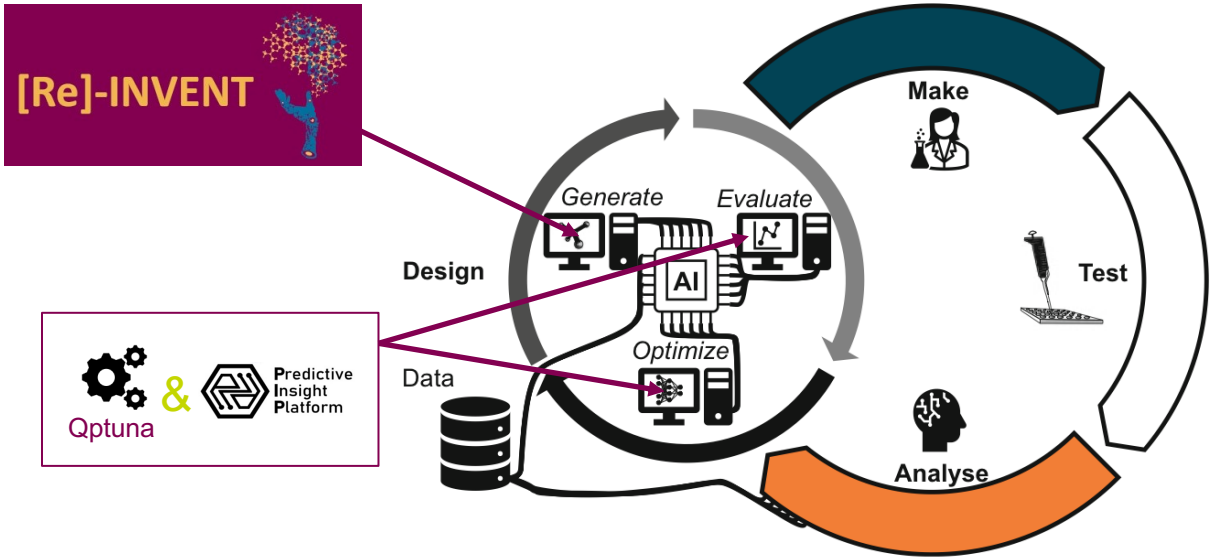
Overview - Future

- Automated model building
- Publication/public release

What is Qptuna?

- QPTUNA: QSAR using Optimization for Hyperparameter Tuning
- Build predictive models for CompChem with hyperparameters optimized by Optuna
- Qptuna library searches for the best ML algorithm and molecular descriptor for the given data
- Automatically apply best practices
- Docs soon to be available

Why?



Thomas M, Boardman A, Garcia-Ortegon M, Yang H, de Graaf C, Bender A. Applications of Artificial Intelligence in Drug Design: Opportunities and Challenges. *Methods Mol Biol*. 2022

Qptuna

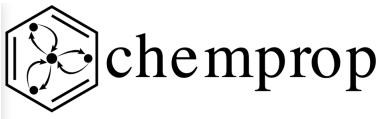
- **Why:**
  - We have assay data, e.g. 1k points.
  - We want a QSAR model to predict assay response for new molecules.
  - Which ML algorithm shall we use? Which molecular representation/fingerprint shall we use?

- **What – looking for the best algorithms and fingerprints:**
  - “Traditional” methods: Grid search, Random search
  - This work: Sequential model-based optimization (SMBO): sequentially construct models to approximate the objective based on historical measurements.

- **How:**
  - Use Optuna library for hyperparameter optimization combined with packages such as Rdkit and e.g. ChemProp

optuna.readthedocs.io/

scikit-learn.org/stable/

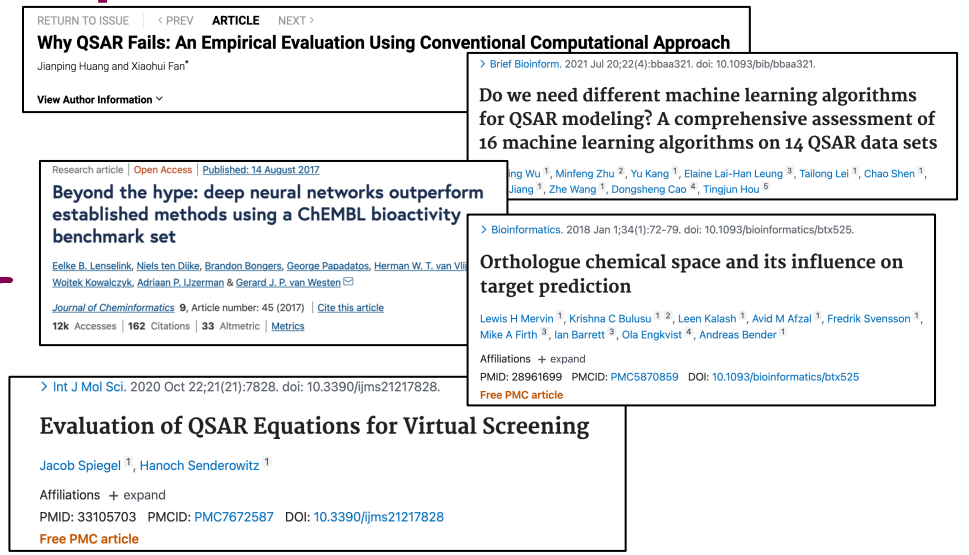


chemprop.readthedocs.io/

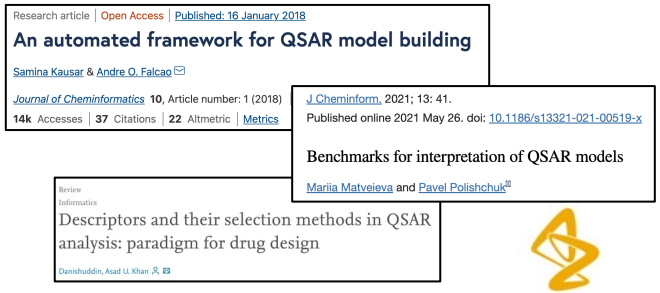
www.rdkit.org/

Available algorithms and descriptors

- **ML Algorithms:**
  - Logistic Regression (aka logit, MaxEnt) classifier
  - SVR (Epsilon-Support Vector Regression)
  - Random Forest
  - Ridge (Linear least squares with L2 regularization)
  - Lasso (Linear model trained with L1 prior as regularizer)
  - Cross decomposition using partial least squares (PLS)
  - XGBregressor (XGBoost: gradient boosting trees algorithm)
  - AdaBoost Classifier SVC (C-Support Vector Classification)
  - SVC (C-Support Vector Classification)
  - ChemProp regression/classification (D-MPNN/FFN)
  - Probabilistic Random Forest

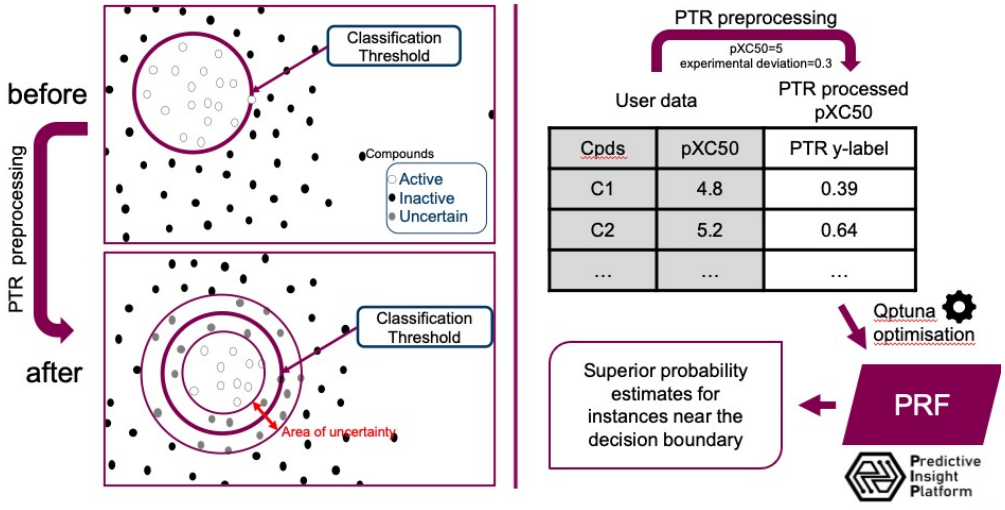


- **Molecular Descriptors:**
  - Avalon (Fingerprint from Avalon Cheminformatics Toolkit)
  - ECFP (Extended Connectivity Fingerprint - MorganFingerprint from RDKit)
  - ECFP_counts (ECFP with counts)
  - MACCS_keys (RDKit SMARTS-based implementation of the 166 public MACCS keys)
  - RDKit Physchem descriptors
  - Special:
    - Scaled Descriptor (takes another descriptor and scales it on training data)
    - Composite Descriptor (concatenates multiple descriptors into one)
    - Precomputed Descriptor from file (reads values from CSV file)
  - SMILES (also side information – related tasks)



Probabilistic Threshold Representation (PTR) learning

- Model the experimental uncertainty in the input data
- Employ best practice
- Somewhere between regression/classification
- Easy to perform. Data transformed and only threshold and Stdev needed



Mervin LH, Trapotsi MA, Afzal AM, Barrett IP, Bender A, Engkvist O. Probabilistic Random Forest improves bioactivity predictions close to the classification threshold by taking into account experimental uncertainty. *J Cheminform*. 2021;13(1):62.

The Qptuna 3-step process

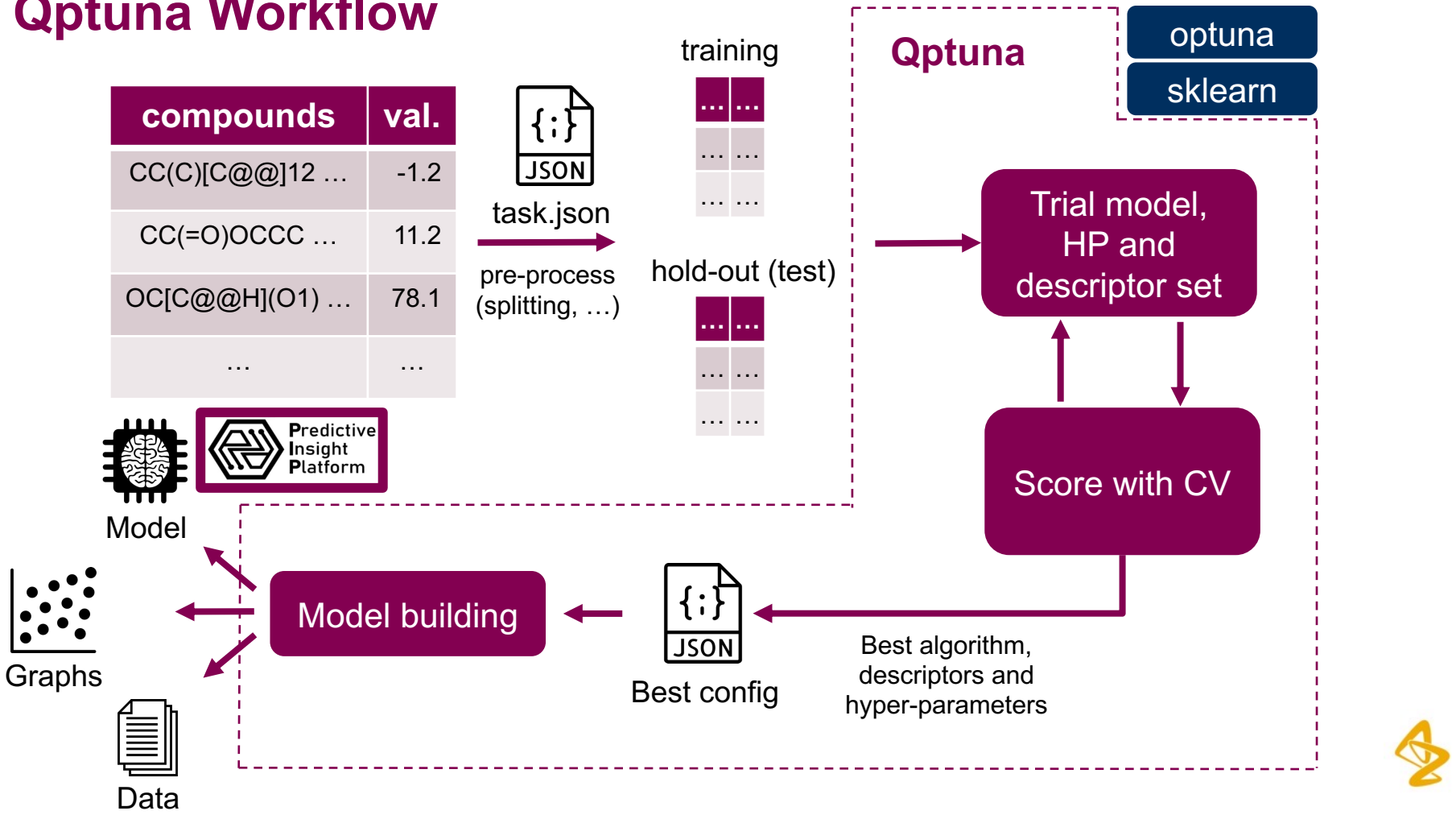
QPTUNA is structured around three steps:

- **Hyperparameter Optimization**: Train many models with different parameters using Optuna. Only the training dataset is used here. Training is done with cross-validation.

- **Build (Training):** Pick the best model from Optimization. Optionally evaluate its performance on the test dataset.

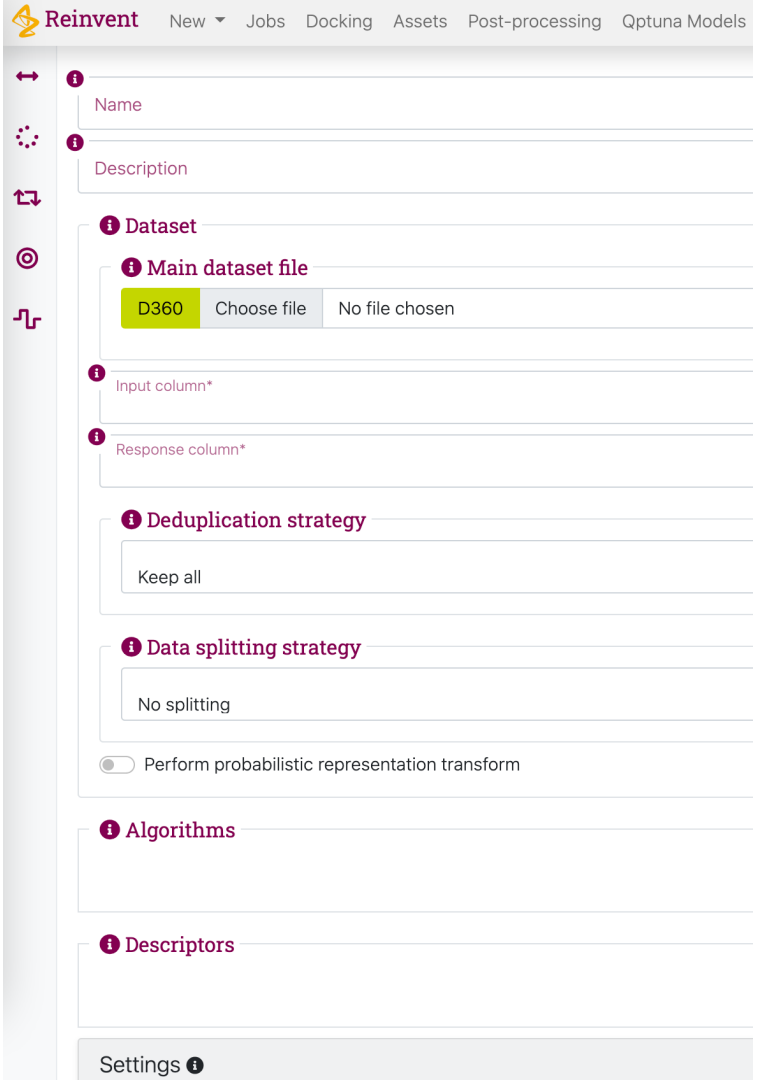
- **“Prod-build:”** Re-train the best-performing model on the merged training and test datasets. Drawback: no data left to evaluate the resulting model, but large benefit that the final model is trained on all available data

Qptuna Workflow



GUI

- GUI for public release available soon
- Improvements to GUI coming soon
- Automatically suggest configurations as good starting point
- Apply sensible defaults to obtain the best model in reasonable timeframe



CLI

- Greater flexibility/customizability
- Less user friendly
- SCP singularity images available
- Local installation possible from bitbucket
- Possible to use run a local CLI/GUI server with MLFlow

[4]:
```
# Prepare hyperparameter optimization configuration.
config = OptimizationConfig(
    data=Dataset(
        input_column="canonical",  # Typical names are "SMILES" and "smiles".
        response_column="molwt",  # Often a specific name (like here), or just "activity".
        training_dataset_file="../tests/data/DRD2/subset-50/train.csv",
        test_dataset_file="../tests/data/DRD2/subset-50/test.csv"  # Hidden during optimiza
    ),
    descriptors=[
        ECFP.new(),
        ECFP_counts.new(),
        MACCS_keys.new(),
    ],
    algorithms=[
        SVR.new(),
        RandomForestRegressor.new(n_estimators={"low": 5, "high": 10}),
        Ridge.new(),
        Lasso.new(),
        PLSRegression.new(),
        XGBRegressor.new(),
    ],
    settings=OptimizationConfig.Settings(
        mode=ModelMode.REGRESSION,
        cross_validation=3,
        n_trials=100,  # Total number of trials.
        n_startup_trials=50,  # Number of startup ("random") trials.
        direction=OptimizationDirection.MAXIMIZATION,
    ),
)
```

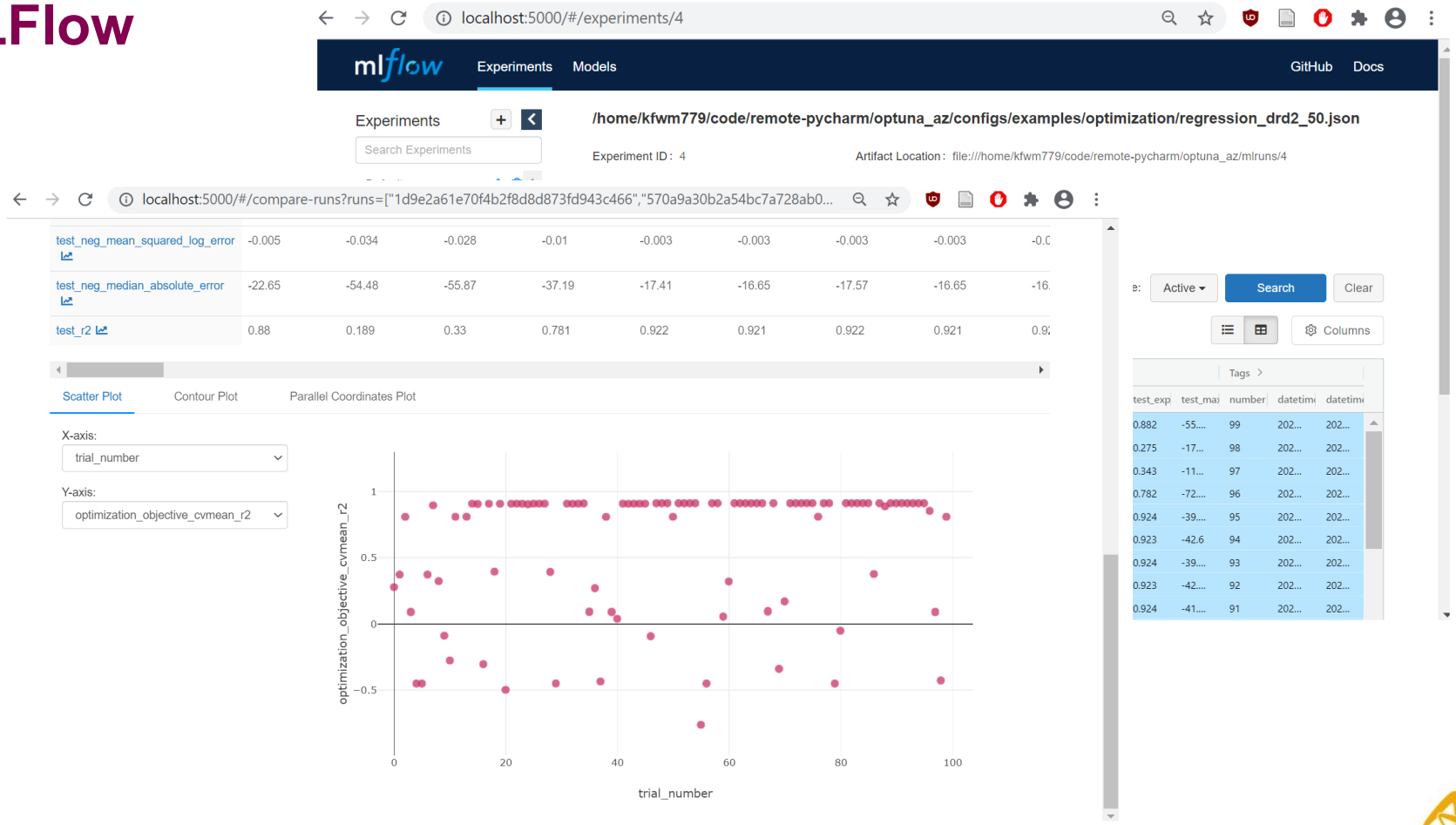
Run optimization

[5]:
```
# Setup basic logging.
import logging
from importlib import reload
reload(logging)
logging.basicConfig(level=logging.INFO)
```

[6]:
```
# Run Optuna Study.
study = optimize(config, study_name="my_study")
# Optuna will log it's progress to sys.stderr
# (usually rendered in red in Jupyter Notebooks).
```

```
[I 2023-01-06 11:00:56,695] A new study created in memory with name: my_study
```

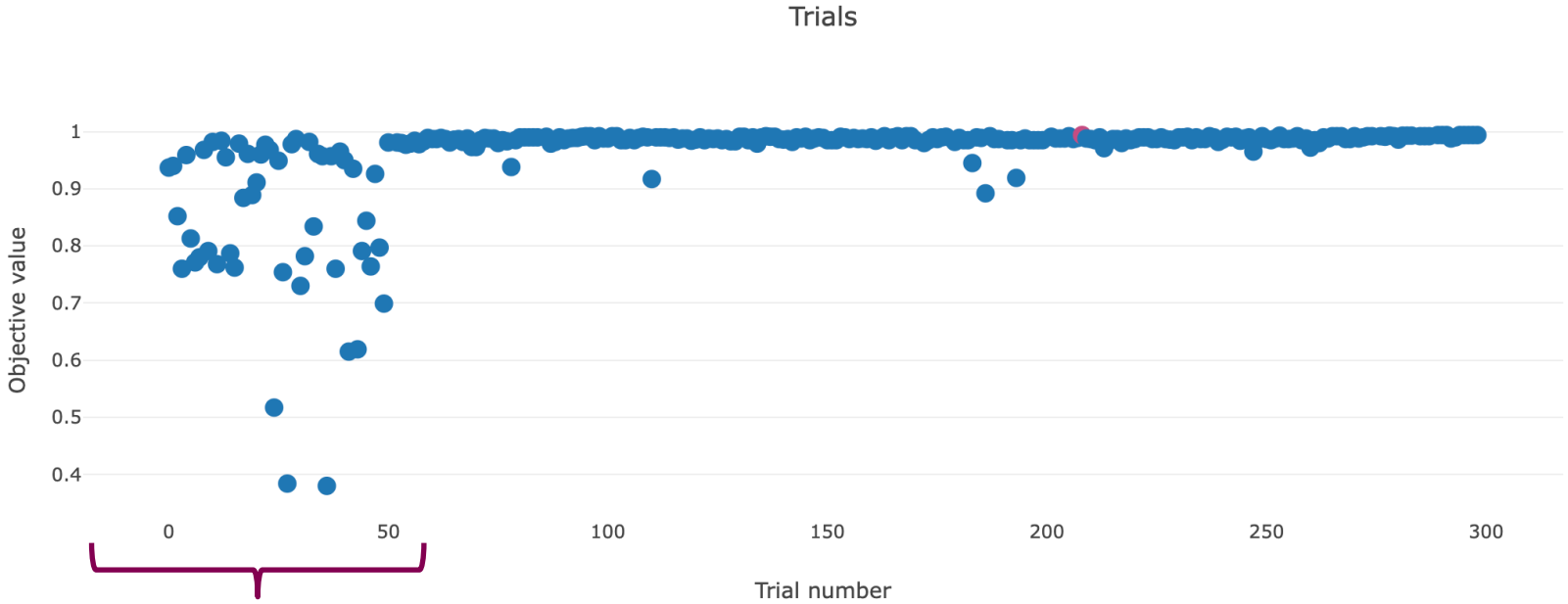

MLFlow


localhost:5000/#/experiments/4
mlflow
Experiments
Models
GitHub
Docs
Experiments
Search Experiments
/home/kfwm779/code/remote-pycharm/optuna_az/configs/examples/optimization/regression_drd2_50.json
Experiment ID: 4
Artifact Location: file:///home/kfwm779/code/remote-pycharm/optuna_az/mlruns/4
localhost:5000/#/compare-runs?runs=["1d9e2a61e70f4b2f8d8d873fd943c466","570a9a30b2a54bc7a728ab0...
test_neg_mean_squared_log_error -0.005 -0.034 -0.028 -0.01 -0.003 -0.003 -0.003 -0.003
test_neg_median_absolute_error -22.65 -54.48 -55.87 -37.19 -17.41 -16.65 -17.57 -16.65
test_r2 0.88 0.189 0.33 0.781 0.922 0.921 0.922 0.921
Active
Search
Clear
Columns
Tags
Scatter Plot
Contour Plot
Parallel Coordinates Plot
X-axis:
trial_number
Y-axis:
optimization_objective_cvmean_r2
optimization_objective_cvmean_r2
trial_number

Optuna settings

- Num trials
- Num of random trials

Pruning now implemented



Real CLS example

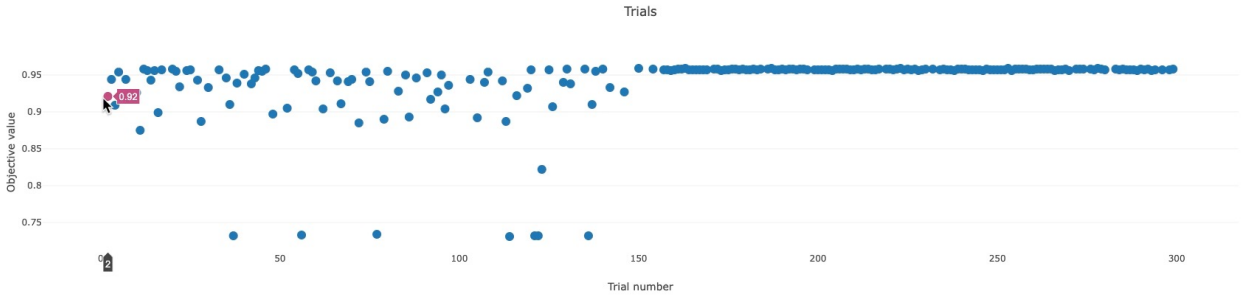
Example Classification

Unpublish job

Slurm Job Id: 38130023

qptuna_optimization job created on 15/02/2023 10:58
slurm status: COMPLETED
internal job id: 63ecbae7933950001247 7f64

Trials | Configuration



Previous jobs

Public

Completed

Example Classification
Project: Qptuna
Description: Default classification set up
User: Nittinger, Eva Nittinger
Slurm job: 38130023
Created: 15/02/2023 10:58

Trial #2

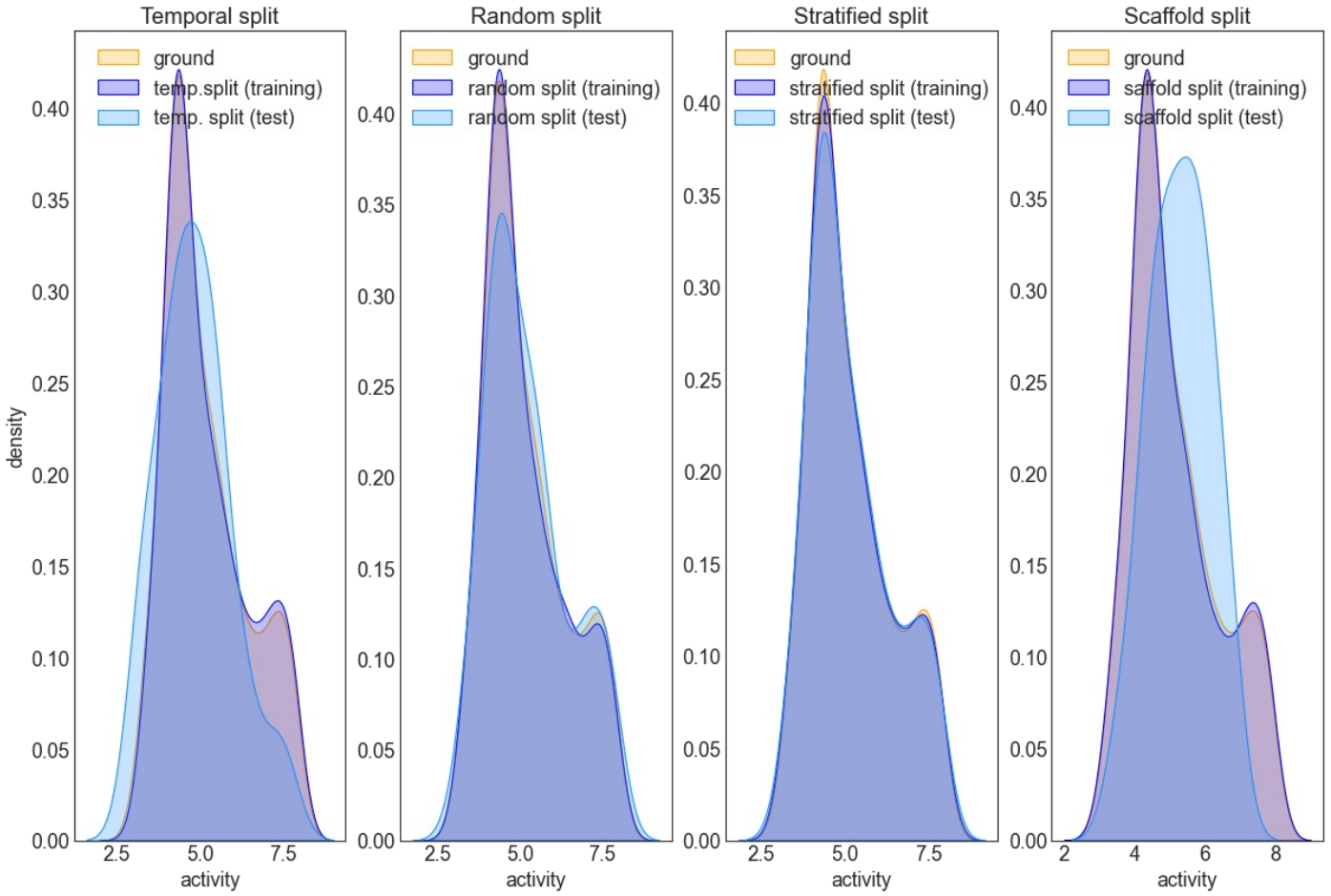
Scoring function	Objective value	Algorithm	Algorithm parameters	Descriptor	Descriptor parameters
roc_auc	0.921	ChemPropHyperoptClassifier	{ "aux_weight_pc": 0, "epochs": 30, "features_generator": "none", "num_iters": 1, "search_parameter_level": "auto" }	SmilesFromFile	{}

Show test scores

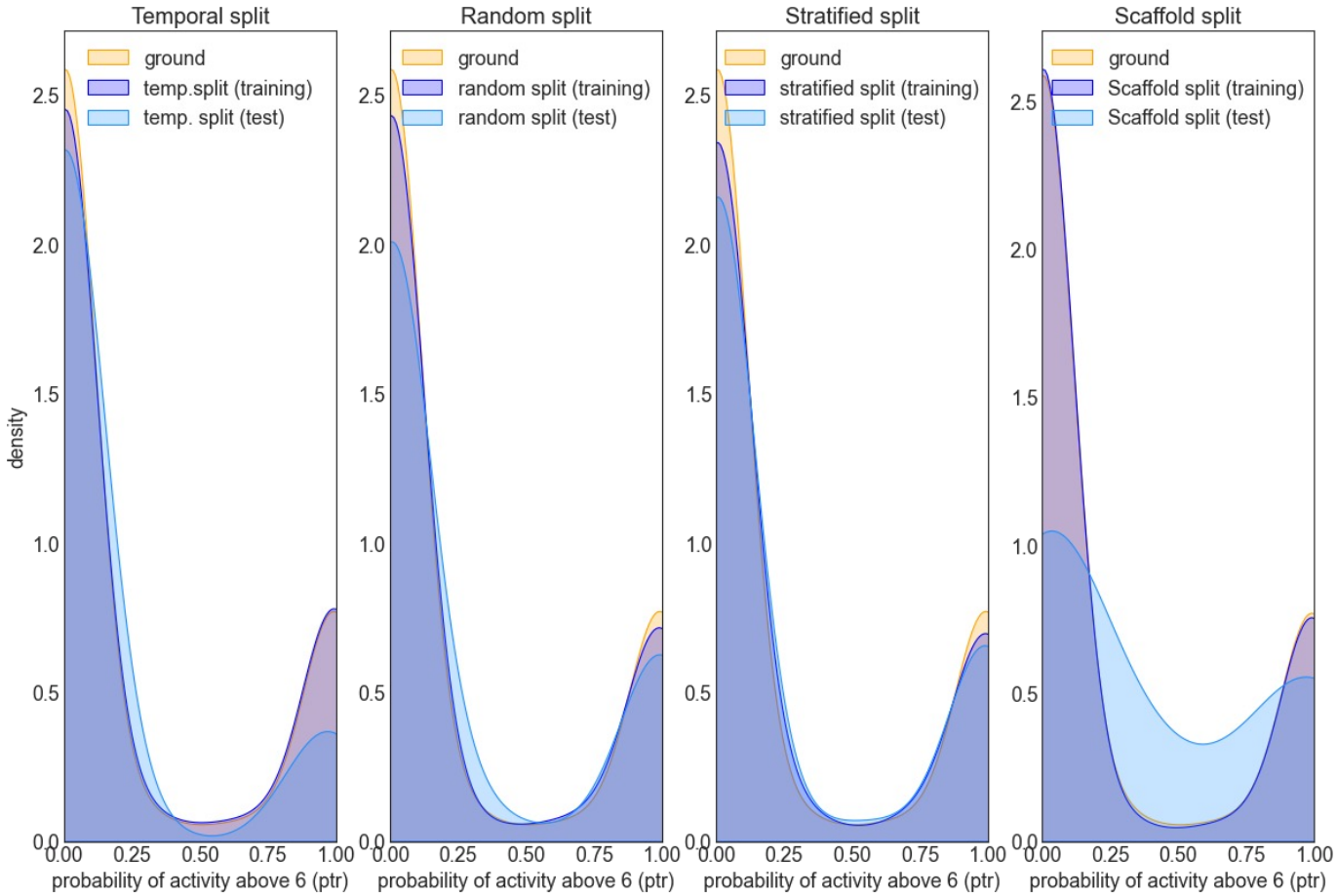
accuracy	average_precision	balanced_accuracy	f1	f1_macro	f1_micro	f1_weighted	jaccard	jaccard_macro	jaccard_micro	jaccard_weighted	precision	precision_macro	precision_micro	precision_weighted	recall	recall_macro	recall_micro	recall_weighted	roc_auc*
0.858	0.859	0.868	0.851	0.857	0.858	0.858	0.741	0.75	0.751	0.752	0.775	0.862	0.858	0.874	0.944	0.868	0.858	0.858	0.919
0.865	0.819	0.874	0.857	0.864	0.865	0.865	0.75	0.761	0.761	0.762	0.785	0.867	0.865	0.879	0.944	0.874	0.865	0.865	0.915
0.861	0.838	0.866	0.848	0.86	0.861	0.862	0.737	0.755	0.756	0.757	0.8	0.859	0.861	0.868	0.903	0.866	0.861	0.861	0.924
0.865	0.832	0.87	0.853	0.864	0.865	0.865	0.743	0.76	0.761	0.763	0.801	0.863	0.865	0.872	0.911	0.87	0.865	0.865	0.911
0.868	0.88	0.881	0.864	0.868	0.868	0.868	0.761	0.767	0.767	0.768	0.776	0.876	0.868	0.89	0.976	0.881	0.868	0.868	0.938

* score was used as objective function

Build (Training): Split for evaluation



PTR example


Temporal split
ground
temp.split (training)
temp. split (test)
Random split
ground
random split (training)
random split (test)
Stratified split
ground
stratified split (training)
stratified split (test)
Scaffold split
ground
Scaffold split (training)
Scaffold split (test)
density
0.0
0.5
1.0
1.5
2.0
2.5
0.00
0.25
0.50
0.75
1.00
probability of activity above 6 (ptr)

Additional features: Uncertainty (Bias)

VennABERS (Classification)

Large-scale probabilistic prediction with and without validity guarantees

Vladimir Vovk, Ivan Petej, and Valentina Fedorova

ROYAL HOLLOWAY UNIVERSITY OF LONDON

Venn-ABERS Predictor

(Preliminary documentation)

The `VennABERS.py` file is a pure Python implementation of the fast Venn-ABERS Predictor described in Vovk2015.

A Venn-ABERS predictor outputs two probability predictions for every test object. In particular, the Venn-ABERS predictor implemented here is the inductive form of probability predictor, which relies on a calibration set. In a nutshell, the Venn-ABERS predictor can be viewed as a distribution-free calibration function that maps scores output by a *scoring classifier* to well-calibrated probabilities. A gentle introduction can be found in this tutorial.

The function that implements the Venn-ABERS Predictor is `ScoresToMultiProbs()`.

```
p0,p1 = ScoresToMultiProbs(calibrPts,testScores)
```

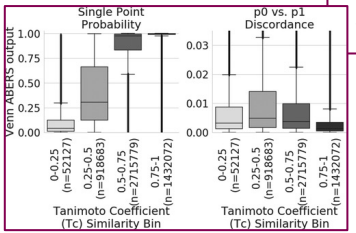
J Chem Inf Model. 2020 Oct 26;60(10):4546-4559. doi: 10.1021/acs.jcim.0c00476. Epub 2020 Sep 21.

Comparison of Scaling Methods to Obtain Calibrated Probabilities of Activity for Protein–Ligand Predictions

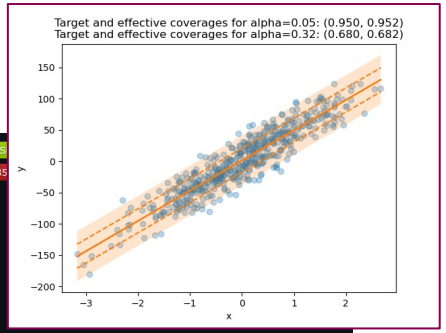
Lewis H Mervin [1], Avid M Afzal [2], Ola Engkvist [3], Andreas Bender [4]

Affiliations + expand

PMID: 32865408 DOI: 10.1021/acs.jcim.0c00476



MAPIE (Regression)



Unit tests passing | codecov 100% | docs passing | license BS
conda-forge v0.6.1 | release v0.6.1 | commits since v0.6.1 0 | 10.485

MAPIE

MAPIE - Model Agnostic Prediction Interval Estimator

MAPIE allows you to easily estimate prediction intervals (or prediction sets) using your favourite scikit-learn-compatible model for single-output regression or multi-class classification settings.

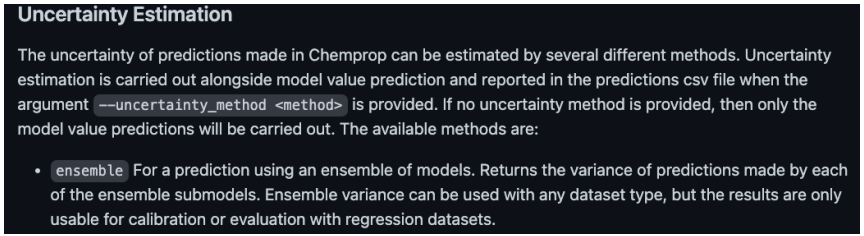
Prediction intervals output by **MAPIE** encompass both aleatoric and epistemic uncertainties and are backed by strong theoretical guarantees thanks to conformal prediction methods [1–7].

Additional features: Uncertainty (variance)

When the data, representation, architecture, etc.
is kept constant.

Only variance is the initialisation

ChemProp (task type agnostic)



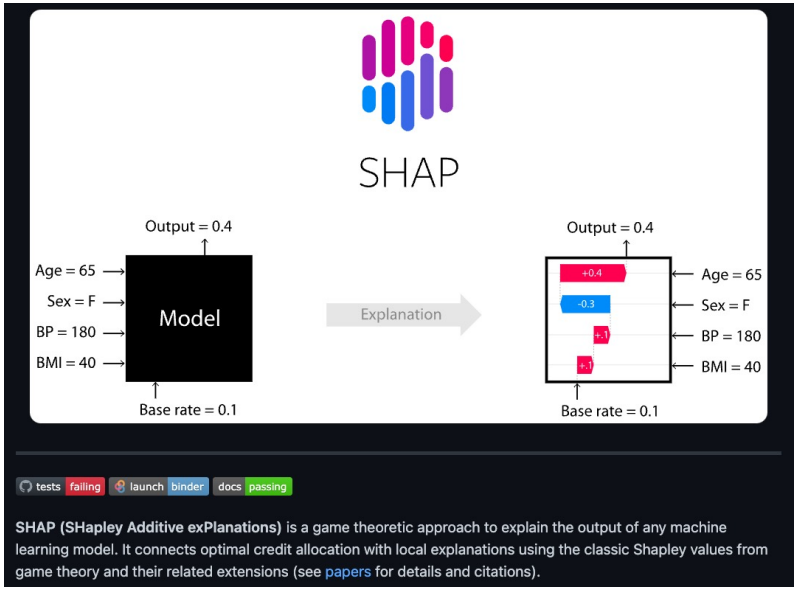
Uncertainty Estimation

The uncertainty of predictions made in Chemprop can be estimated by several different methods. Uncertainty estimation is carried out alongside model value prediction and reported in the predictions csv file when the argument `--uncertainty_method <method>` is provided. If no uncertainty method is provided, then only the model value predictions will be carried out. The available methods are:

- `ensemble` For a prediction using an ensemble of models. Returns the variance of predictions made by each of the ensemble submodels. Ensemble variance can be used with any dataset type, but the results are only usable for calibration or evaluation with regression datasets.

Additional features: Explainability

Shallow models



- Auto select appropriate explainer (Tree/Linear/Permutation) for compatible algorithms
- Model agnostic example with KernelExplainer to explain unsupported algorithms
- Output: importance for features in a chemical series/ input
- ECFP's explained for bits turned on

ChemProp

Interpreting

It is often helpful to provide explanation of model prediction (i.e., this molecule is toxic because of this substructure). Given a trained model, you can interpret the model prediction using the following command:

```
chemprop_interpret --data_path data/tox21.csv --checkpoint_dir tox21_checkpoints/fold_0/ --property_
```

If installed from source, `chemprop_interpret` can be replaced with `python interpret.py`.

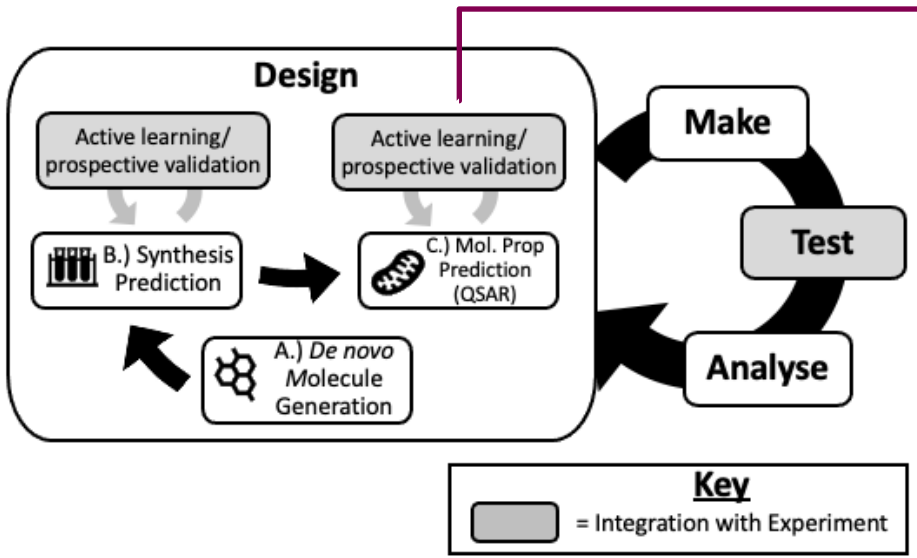
The output will be like the following:

- The first column is a molecule and second column is its predicted property (in this case NR-AR toxicity).
- The third column is the smallest substructure that made this molecule classified as toxic (which we call rationale).
- The fourth column is the predicted toxicity of that substructure.

As shown in the first row, when a molecule is predicted to be non-toxic, we will not provide any rationale for its prediction.

smiles	NR-AR	rationale	rationale_score
(C(F)(F)F)cc([N+](=O)[O-])c1Cl	0.014		
]2C[C@H]3[C@@H]4C[C@H] =C[C@]5(C)[C@H]4[C@@H])(C)[C@]2(C(=O)CO)O1	0.896	C[C@]12C=CC(=O)C=C1[CH2:1]C[CH2:1] [CH2:1]2	0.769
2CC[C@H]3[C@@H])@H]4C(O)=C(C#N)C[C@]35C))H]1CC[C@@H]2O	0.941	C[C@]12C[CH:1]=[CH:1] [C@H]3O[C@]31CC[C@@H]1[C@@H]2CC[C:1] [CH2:1]1	0.808
:@H](O)[C@H]3[C@@H] :C(=O)CC[C@@]43C))2(O)C(=O)COP(=O)([O-])[O-]	0.957	C1C[CH2:1][C:1] [C@@H]2[C@@H]1[C@@H]1CC[C:1] [C:1]1C[CH2:1]2	0.532

Qptuna to enable AL integration within the DMTA cycle



Fill up plates with compounds from PLS that are predicted to improve the model (uncertainty/performance)

Suggest compounds using AL

Janet JP, Mervin L, Engkvist O *"Artificial Intelligence in Molecular de novo Design - Integration with Experiment"* Current Opinion in Structural Biology (accepted)

Live demo

Future Plans

- GUI improvements
- Documentation improvements
- Active learning

Confidentiality Notice