DEEL

DEpendable & Explainable Learning

PUNCC: A Python Library for Predictive UNcertainty Calibration and Conformalization

Mouhcine Mendil, Luca Mossina, David Vigouroux

IVADO

IRT SAINT EXUPÉRY

CRIAQ
Consortium de recherche et d'innovation en aérospatiale au Québec

ANITI
Porté par l'Université de Toulouse

Institut intelligence et données

Agenda

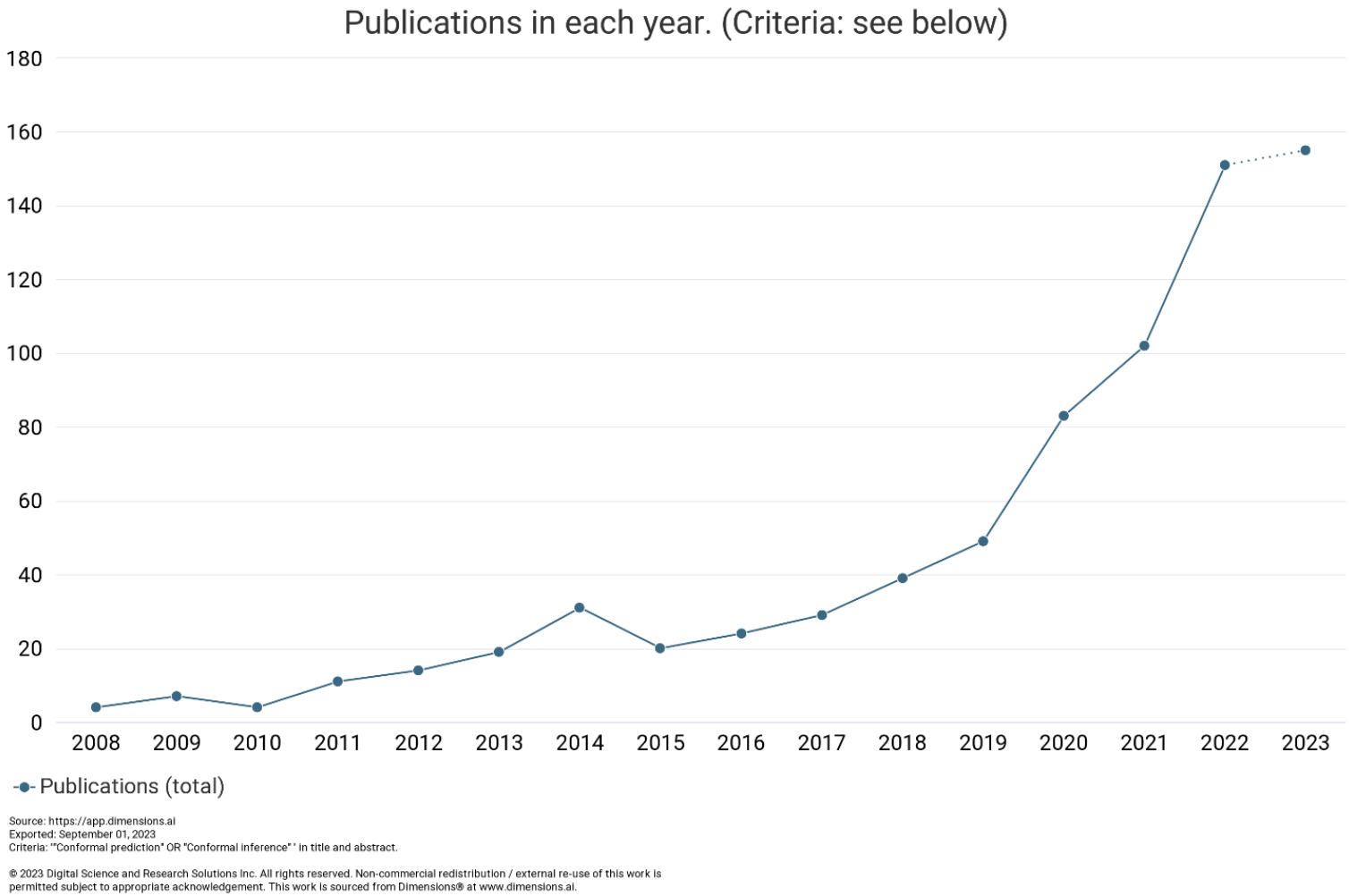
Introduction

Properties of PUNCC

Experiments

Conclusion

Introduction: context

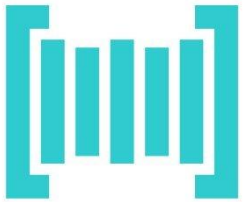


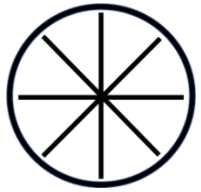
Introduction: context

- ❑Conformal prediction
- ❑Nonexchangeable conformal prediction
- ❑Conformal risk control
- ❑Conformal training
- ❑…

Introduction: context

Lively open source ecosystem (check awesome-CP for more refs):





Challenge: maturity vs upgradability

Introduction: PUNCC

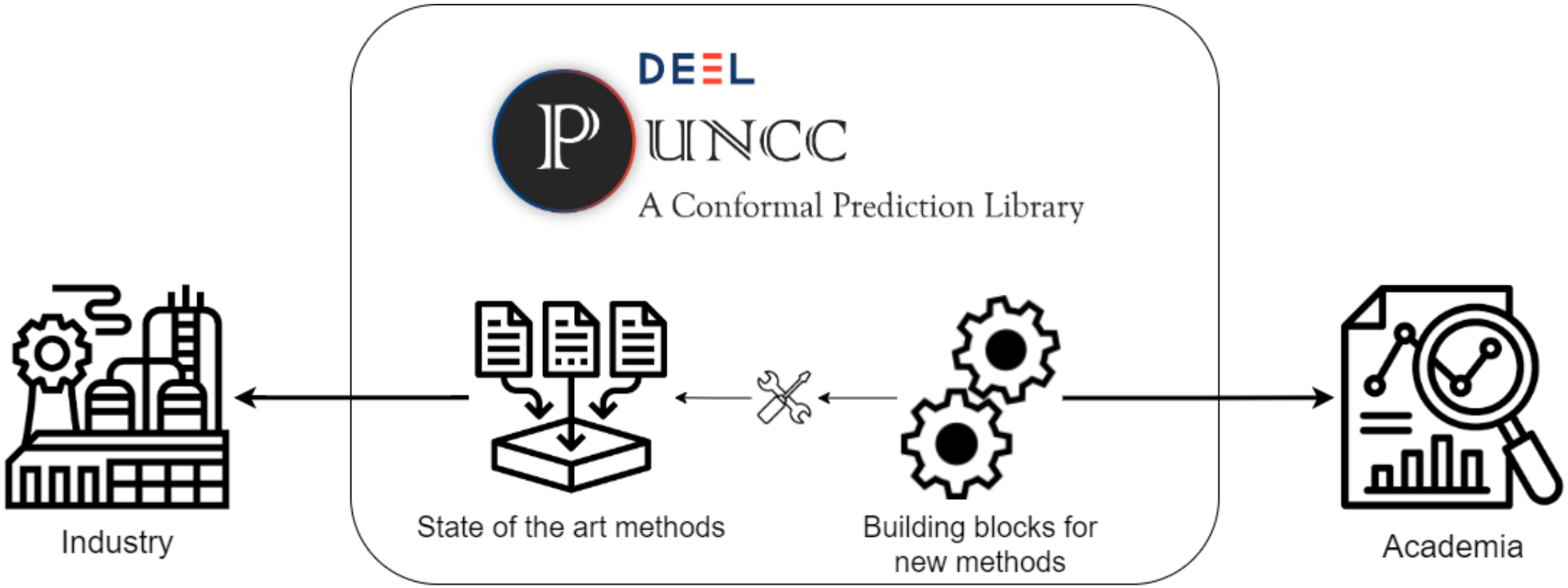
DEEL
DEpendable & Explainable Learning

DEEL

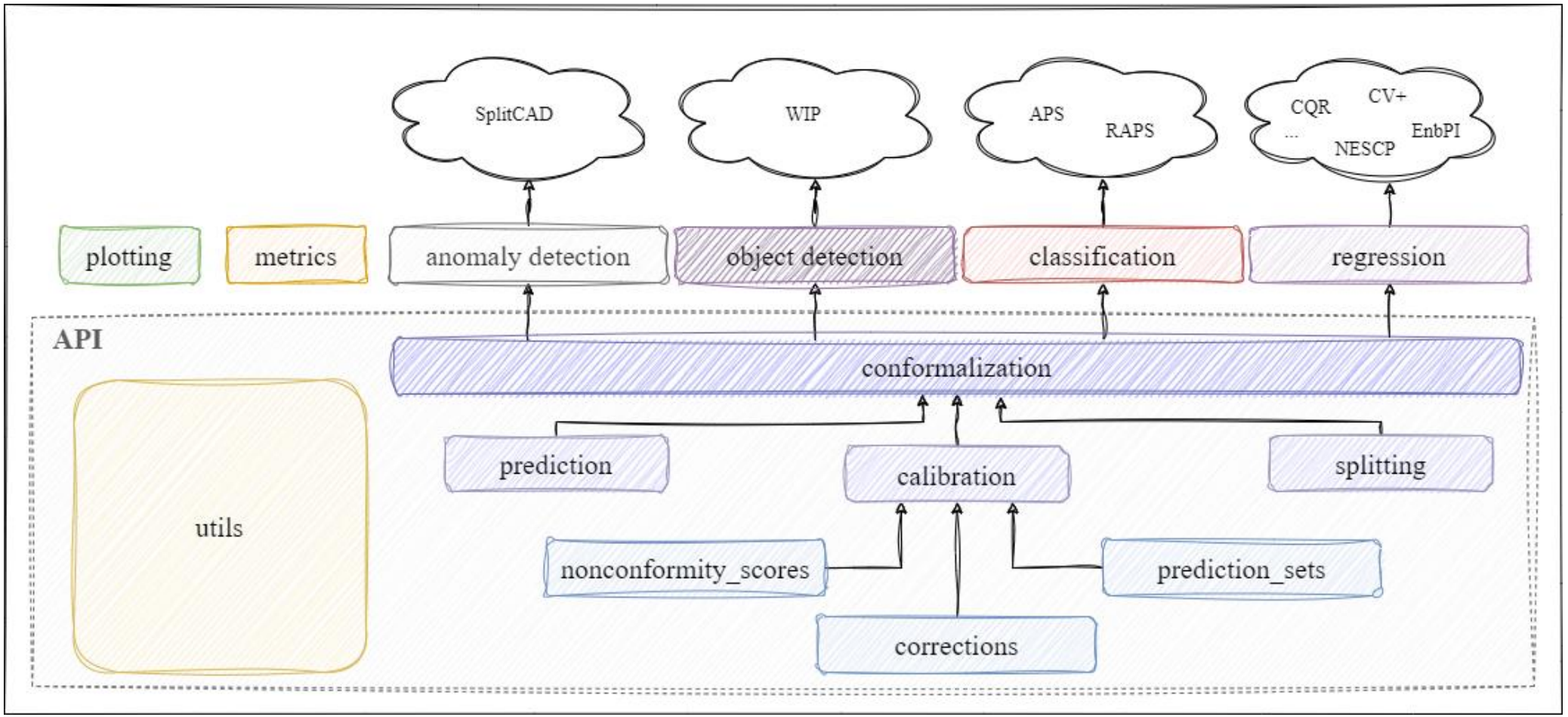
PUNCC

A Conformal Prediction Library

Introduction: PUNCC



Introduction: PUNCC



DEEL

DEpendable & Explainable Learning

Properties of PUNCC

- User-friendliness
- Simplicity
- Interoperability
- Extensivity
- Flexibility

IVADO

IRT SAINT EXUPÉRY

CRIAQ
Consortium de recherche et d'innovation en aérospatiale au Québec

ANITI
Porté par l'Université de Toulouse

Institut intelligence et données

Properties of PUNCC: User-friendliness

- ❑Documentation
- ❑Tutorials
- ❑Tests
- ❑Updates and maintenance

Properties of PUNCC: Simplicity

DEEL
DEpendable & Explainable Learning

1

```python
from deel.puncc.regression import SplitCP, CQR, EnbpI

# ICP
cp_alg = SplitCP(predictor)

# CQR
cp_alg = CQR(predictor)

# EnbPI
cp_alg = EnbPI(predictor, B=30)
```

```python
from deel.puncc.classification import APS

cp_alg = APS(predictor)
```

```python
from deel.puncc.anomaly_detection import SplitCAD

cp_alg = SplitCAD(predictor)
```

2

```python
# Proper training and calibration
cp_alg.fit(X, y)
```

3

```python
# Inference
preds = cp_alg.predict(X_new, alpha)
```

- ❑ **Consistent interface and workflow:**

1. Initialization
2. Proper training and calibration
3. Inference

- ❑ **Many possible variations (pretrain, calibration, hyper-parameters…)**

Properties of PUNCC: Interoperability

DEEL
DEpendable & Explainable Learning

- Support most data types and ML libraries

```python
from deel.puncc.api.prediction import BasePredictor

# Definition of a predictor
my_predictor = BasePredictor(my_model)
```

- Natively support Scikit-learn, TensorFlow, PyTorch, XGBoost, …

- Can work on top of UQ libraries too

Properties of PUNCC: Extensivity

- Several sota methods implemented covering different tasks

Conformal Method	Source	Task
Split Conformal Prediction (SCP)	Papadopoulos et al. (2002)	Regression
Locally Adaptive CP (LACP)	Lei et al. (2018)	Regression
Conformalized Quantile Regression (CQR)	Romano et al. (2019)	Regression
Cross-Validation+ (CV+)	Barber et al. (2019)	Regression
Adaptive Prediction Sets (APS)	Romano et al. (2020)	Classification
Regularized Adaptive Prediction Sets (RAPS)	Angelopoulos et al. (2020)	Classification
Ensemble batch Prediction Interval (EnbPI)*	Xu and Xie (2021)	Regression
*Weighted Split Conformal Prediction (WSCP)**	Barber et al. (2022)	Regression
Inductive Conformal Anomaly Detection (ICAD)	Laxhammar et al. (2015)	Anomaly detection

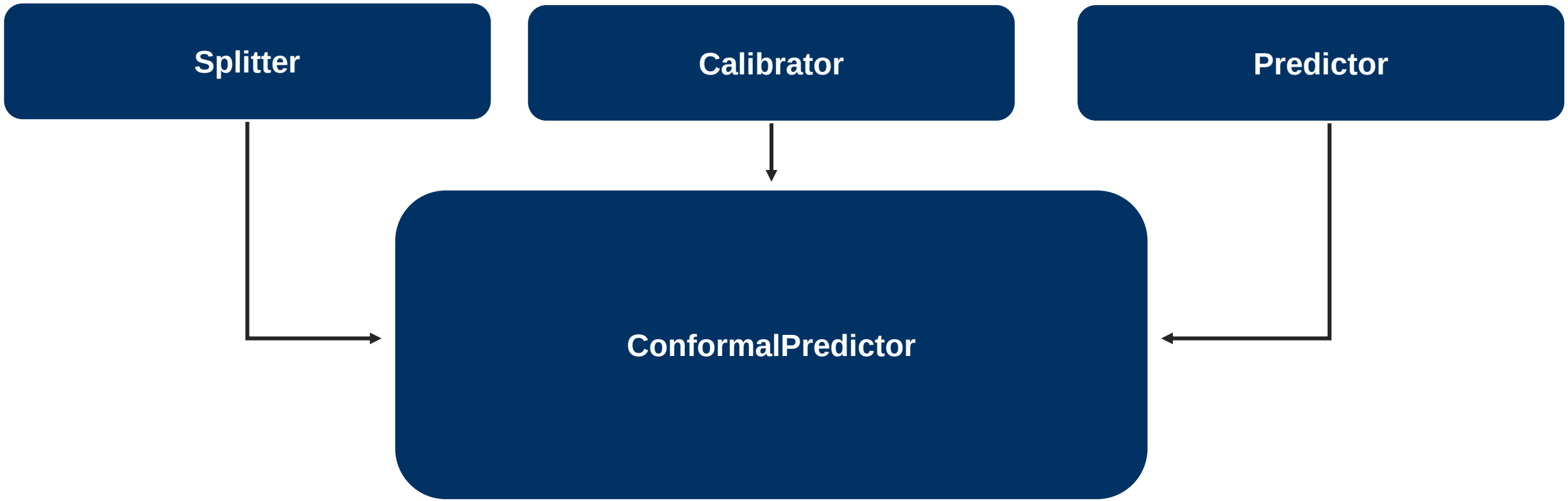
- Community oriented: Contributors can easily share their implementations

Properties of PUNCC: Flexibility

- Easily design your (own) conformal predictors with the API
  - Custom data attribution for proper training and calibration
  - New nonconformity scores
  - New ways of computing prediction sets
  - (New approaches to adaptively update the sensitivity level)
- No need to code everything from scratch, just implement what's new
- Rely on tested and maintained code to build and experiment new algorithms

Properties of PUNCC: Flexibility

- Example: weighted JKK+ CQR



Properties of PUNCC: Flexibility

DEEL
DEpendable & Explainable Learning

❑Example: weighted JKK+ CQR

```python
from deel.puncc.api.splitting import KFoldSplitter

# LOO data schemes
cv_splitter = KFoldSplitter(K=n, random_state=0)
```

```python
from deel.puncc.api.calibration import BaseCalibrator
from deel.puncc.api.nonconformity_scores import cqr_score
from deel.puncc.api.prediction_sets import cqr_interval

# Define a calibrator from the API
calibrator = BaseCalibrator(nonconf_score_func=cqr_score,
                            pred_set_func=cqr_interval,
                            weight_func=exp_decay)
```

```python
from deel.puncc.api.prediction import DualPredictor

# Wrap qunatile regressors in predictor
gb_predictor = DualPredictor(models=[regressor_q_low,
                                     regressor_q_hi])
```

```python
from deel.puncc.api.conformalization import ConformalPredictor

# Instanciate a conformal prediction object
cp_alg = ConformalPredictor(predictor=gb_predictor,
                            calibrator=calibrator,
                            splitter=cv_splitter,
                            train=True)

# Proper training and calibration
cp_alg.fit(X_train, y_train)

# Conformal inference
preds = cp_alg.predict(X_test, alpha=.1)
```

DEEL

DEpendable & Explainable Learning

Experiments

IVADO

IRT SAINT EXUPÉRY

CRIAQ

Consortium de recherche et d'innovation en aérospatiale au Québec

ANITI

Porté par l'Université de Toulouse

Institut intelligence et données

Experiments

- Experiments using PUNCC

Datasets	Data types	Underlying Models	ML Libraries	CP Methods
California housing prices	Tabular	MLP Xgboost Gradient boosting Nearest neighbors	TensorFlow Pytorch XGBoost Sklearn	SCP LACP CQR/CV+
Imagenette Imagewoof	Images	ResNet-50	TensorFlow	APS RAPS
Elec2	Time series	Linear	Sklearn	OSSCP NESCP EnbPI

Experiments: Metrics

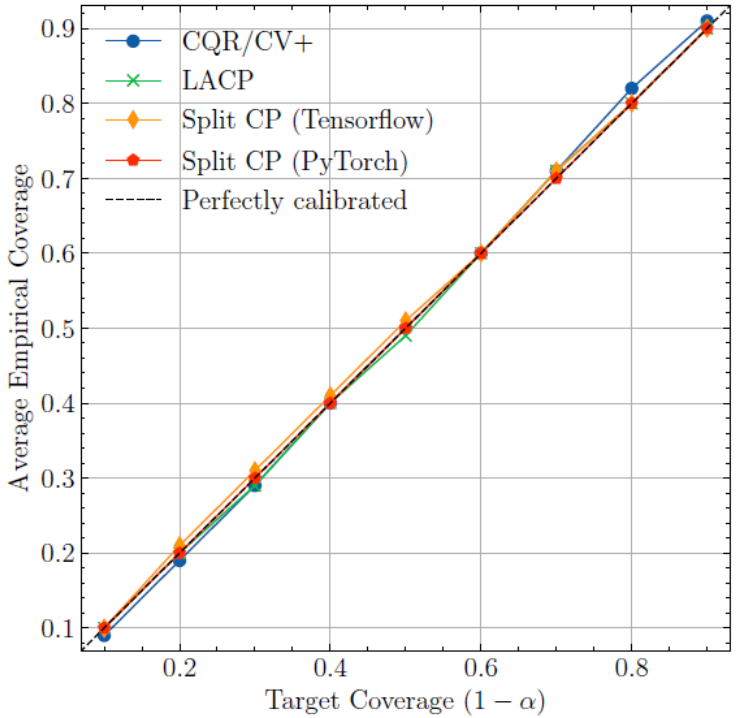


Figure 3: Average coverage gap (across 20 validation folds).

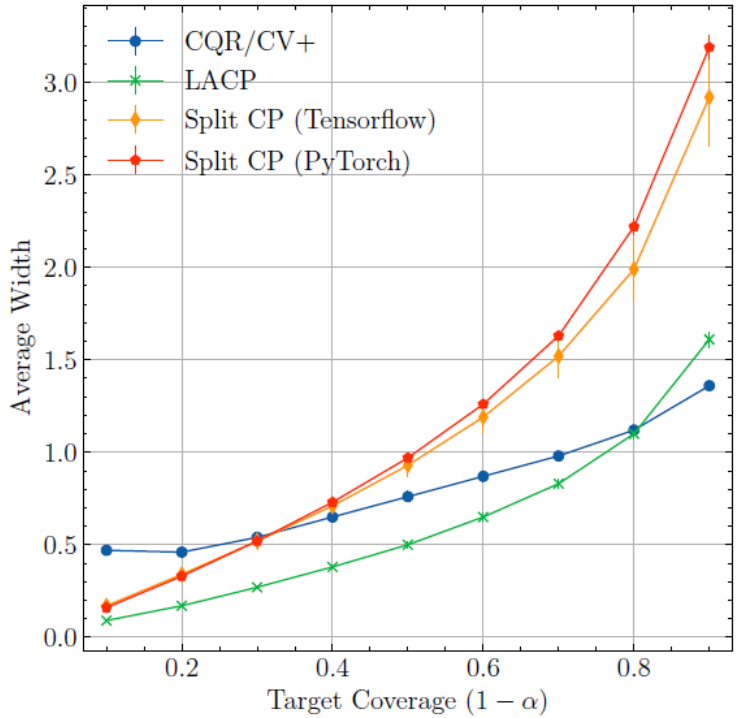


Figure 4: Average empirical size of PIs (across 20 validation folds).

Experiments: Plotting

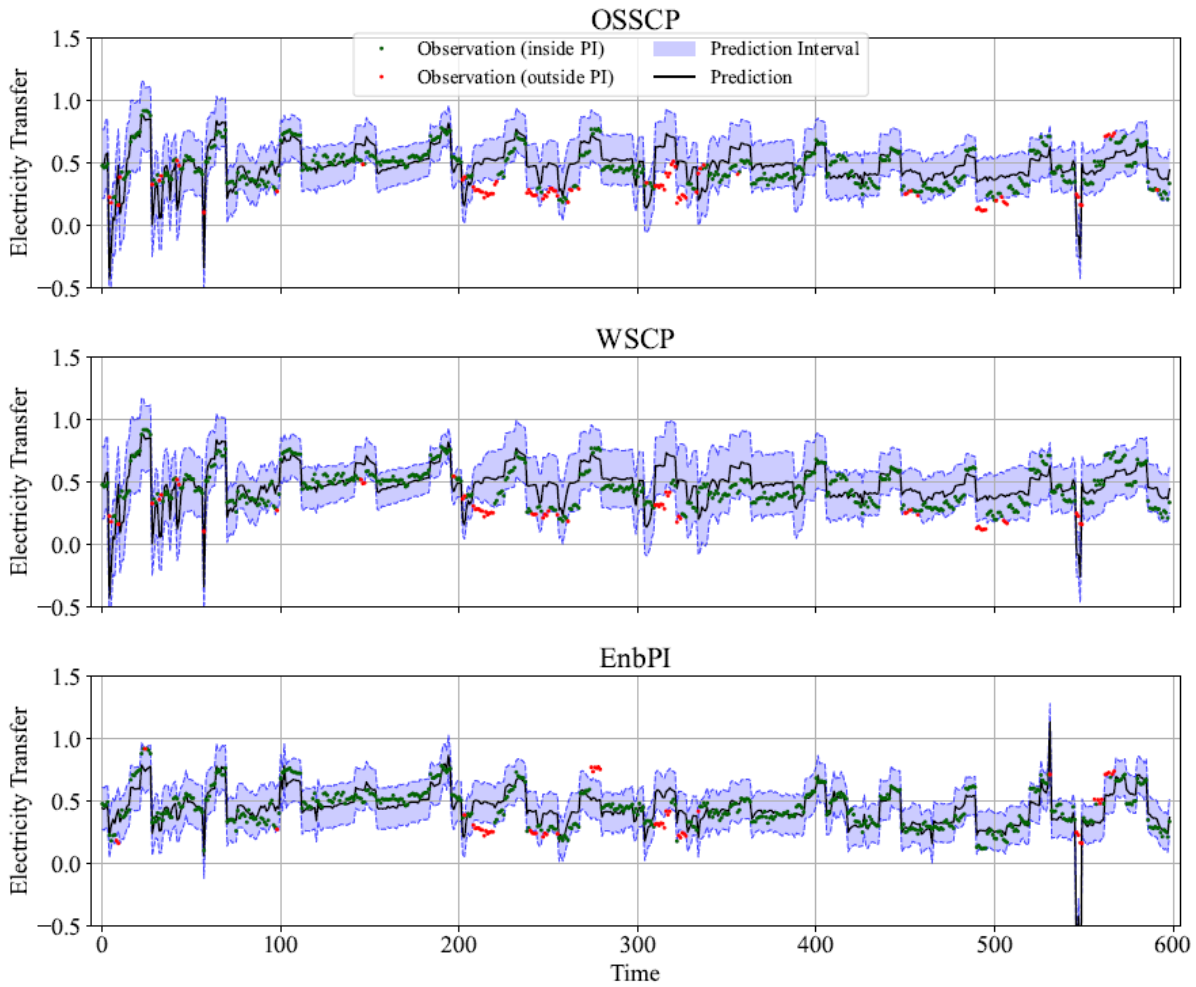


Figure 9: PIs on 600 test samples for three different algorithms (using `puncc.plotting`).

Conclusion

- Open-source library in Python for uncertainty quantification based on CP
- Interoperable with most data types and ML frameworks
- Collection of state of the art CP-related algorithms
- Consistent workflow to facilitate learning and benchmarks
- API to have full control on the design of new/advanced CP procedures

Perspectives

- ❑Team is expanding !
- ❑More sota methods
- ❑Conformal multivariate regression (object detection)
- ❑Further projects

DEEL
France-Quebec

https://github.com/deel-ai/puncc

IVADO

IRT SAINT EXUPÉRY

CRIAQ
Consortium de recherche et d'innovation en aérospatiale au Québec

ANITI
Porté par l'Université de Toulouse

Institut intelligence et données

Properties of PUNCC: Interoperability

- Easy to use wrapper to conform to API constraints

Example:

RAPS on RandomForestClassifier, or CatBoostClassifier need logit predictions

```python
from sklearn.ensemble import RandomForestClassifier
from deel.puncc.api.prediction import BasePredictor

# Create a random forest classifier
rf_model = (n_estimators=100, random_state=0)

# Create a wrapper of the random forest model
# to redefine its predict method into logits predictions
class RFPredictor(BasePredictor):
    def predict(self, X, **kwargs):
        return self.model.predict_proba(X, **kwargs)

# Wrap model in the newly created RFPredictor
rf_predictor = RFPredictor(rf_model)
```

- Can work on top of UQ libraries too