Madrid

"Prediction using Conformal Prediction"

Carl McBride Ellis

- LinkedIn www.linkedin.com/in/carl-mcbride-ellis
- kaggle www.kaggle.com/carlmcbrideellis
- carl.mcbride@protonmail.ch

(v 1.0.0)

Conformal prediction is a framework designed to provide **valid uncertainty quantification**

- the math: somewhat complicated
- the usage: ¡really easy!

We shall show exactly how to apply CP to

- regression $\rightarrow$ better prediction intervals
- binary classification $\rightarrow$ well calibrated probabilities

Regression

In ML (RandomForest, XGBoost, *etc*) we usually create point predictions (expectation values):

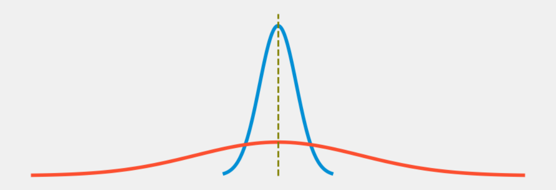
$$\mathbb{E}(Y|X = x)$$

For the L2 loss function $\mathbb{E}$ will be an estimate of the mean, $\hat{\mu}(x)$

The mean is a measure of central tendency.

Ok ¿so what is the problem?

These two distributions have exactly the same mean value:



— point predictions tell us nothing about the dispersion

Motivation:

House price: model (point) prediction = **121,516€**

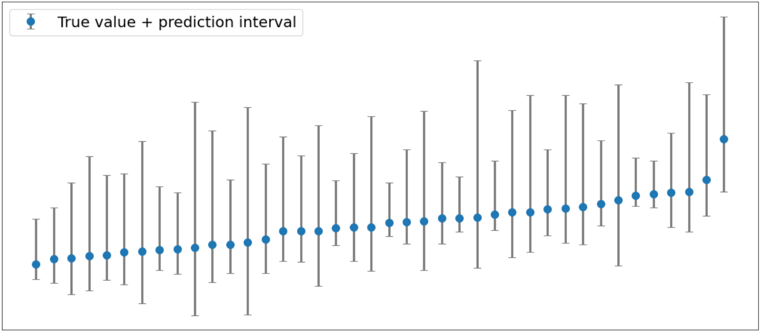
- If we saw a house with the very same characteristics on sale for **120,000€** *¿is that a bargain?*

- If we put an asking price of **125,000€** on the house *¿will it sell, or is it overpriced?*

This is where prediction intervals can really help us to make a **decision**.

We can specify a coverage (say 80%) and then produce a model. That model could return a lower price of say 110,000€ and an upper price of 130,000€ for said house.

With this data we are now confident that if we see a house with the same characteristics on sale for 100,000€ then this property is below market value, and could be a very good investment opportunity. Or on the other hand we could confidently try to sell the property at 130,000€ without leaving money on the table.



prediction intervals $\neq$ error bars
prediction intervals $\neq$ confidence intervals

We shall describe the Conformalized Quantile Regression (CQR) technique by Emmanuel Candès and co-workers in 2019

Romano, Patterson, Candès "*Conformalized Quantile Regression*", NeurIPS **vol 32** (2019)

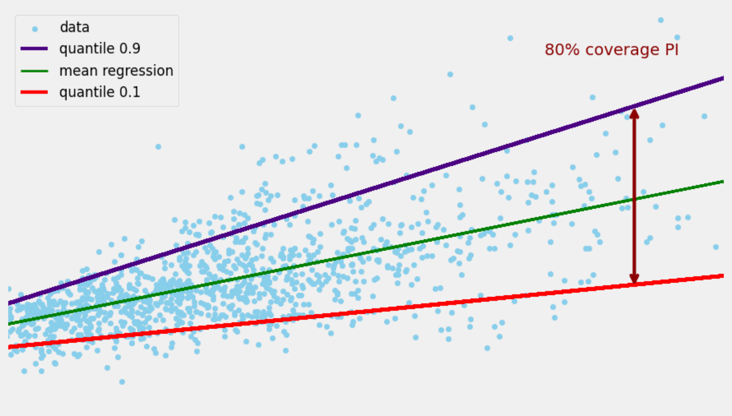
Sesia, Candès "*A comparison of some conformal quantile regression methods*", Stat **vol 9** e261 (2020)

which adds CP to standard quantile regression

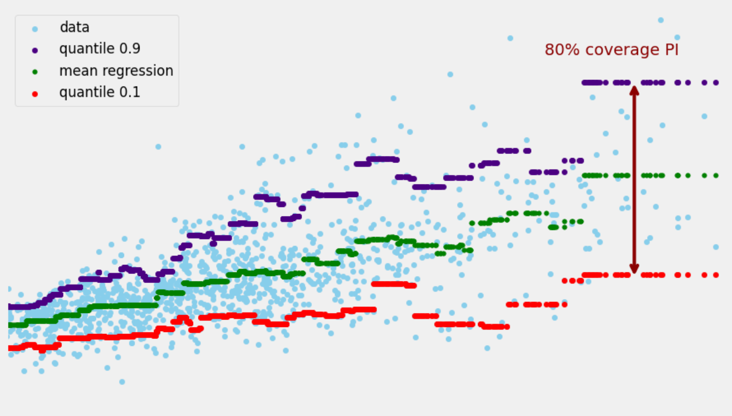
Bassett, Koenker "*Regression Quantiles*", Econometrica **vol 46** pp. 33-50 (1978)

Koenker, Hallock "*Quantile Regression*", Journal of Economic Perspectives **vol 15** pp. 143-156 (2001)

What does quantile regression look like? (linear model)



...and for a non-parametric tree model



Some ways to perform quantile regression in python:

- `sklearn.linear_model.QuantileRegressor`
- `sklearn.ensemble.GradientBoostingRegressor`
- `sklearn.ensemble.HistGradientBoostingRegressor`
- `XGBoost` / `LightGBM` / `CatBoost`

There is also a new python `quantile-forest` package for quantile regression forests, written by Reid Johnson of the Zillow Group

These can be used as our 'base' estimator

Ok, ¿so what is the problem?

...the intervals from our base estimator may be wrong!

we may ask for 80% coverage, *but we may not get 80% coverage*

Conformal prediction will guarantee our specified coverage.

There are two types of coverage: **marginal** and **conditional**

For, say 80% coverage:

- **marginal**: 80% of the intervals will contain `y_true`
- **conditional**: a given interval has an 80% chance of containing `y_true`

It is not possible to guarantee distribution-free conditional coverage.
However, CP does guarantee marginal coverage
- this is what is meant by **validity**.

Barber, Candès, Ramdas, Tibshirani "*The limits of distribution-free conditional predictive inference*", arXiv:1903.04684 (2020)

We want our prediction intervals to have 3 things:

- **validity** — guaranteed marginal coverage
- **efficiency** — intervals to be as narrow as possible
- **adaptability** — in cases of heteroscedasticity have variable lengths

- conformal prediction provides the validity
- the base estimator provides the efficiency

  (validity can also help the efficiency; overcoverage is inefficient)
- the quantile regression in CQR provides the adaptability

Zecchin, Park, Simeone, Hellström "*Generalization and Informativeness of Conformal Prediction*", arXiv:2401.11810 (2024)

CP comes in two ‘flavours’:

- **transductive** (“full” or “online”)

  no calibration set needed, but computationally expensive: multiple fits

- **inductive** (“split” or `"prefit"`)

  needs hold-out data for a calibration set, but is computationally efficient: just one fit

Requirement:

For ML to work correctly our splits should be i.i.d. samples

For CP to work we need *exchangeability*

¿What is exchangeability?

It is when one can `shuffle` or `sort` the rows of a dataset and it will make no difference to the predictions.

(Note: This becomes problematic for time series data, where the order of the rows is indeed important)

Exchangeability is a weaker condition than iid,
— so if we are iid we are good to go!

Kuchibhotla "*Exchangeability, Conformal Prediction, and Rank Tests*", arXiv:2005.06095 (2021)

CQR: 6 step process:

1) train your two quantile regressors, Q_{lower} and Q_{upper}, on `X_train`, `y_train`
2) predict the two quantiles for the `X_calibrate` data
3) for each `y_true` in `y_calibrate` calculate a **conformity score** E_i

$$E_i := \max(Q_{lower} - \texttt{y\_true},\ \texttt{y\_true} - Q_{upper})$$

4) calculate `s` being the value of the $(1-\alpha)(1+1/n_{calib})$ quantile of this set of conformity scores $\{E\}$

   ($\alpha :=$ (100-coverage%)/100, and n_{calib} is the number of calibration rows)

5) predict the two quantiles for `X_test`
6) obtain **validity** by subtracting `s` from Q_{lower} and adding `s` to Q_{upper}

MAPIE

MAPIE - Model Agnostic Prediction Interval Estimator

```
!pip install -q mapie
```

- GitHub: `scikit-learn-contrib/MAPIE`
- readthedocs

Set up the split datasets (*i.e.* train/calibration/test)

```
from sklearn.model_selection import train_test_split
X_train, X_tmp,  y_train, y_tmp
            = train_test_split(X, y, test_size=2000)
X_calib, X_test, y_calib, y_test
            = train_test_split(X_tmp,y_tmp,test_size=1000)
```

choose a base quantile estimator

```
from lightgbm import LGBMRegressor
base_reg = LGBMRegressor( objective = "quantile",
                          alpha     = alpha )
```

```
from mapie.regression import MapieQuantileRegressor

mapie = MapieQuantileRegressor( estimator=base_reg,
                                cv="split",
                                alpha=0.2) % 80% coverage

mapie.fit(X_train, y_train,
          X_calib=X_calib, y_calib=y_calib)

y_pred, y_intervals = mapie.predict(X_test)
```

`y_pred` contains our point predictions, and
`y_intervals` contains our lower and upper intervals.

Prediction interval performance metric: Mean Winkler interval score

$$W_\alpha = \begin{cases} (u-l) + \frac{2}{\alpha}(l-y), & \text{if } y < l \\ (u-l), & \text{if } l \leq y \leq u \\ (u-l) + \frac{2}{\alpha}(y-u), & \text{if } y > u \end{cases}$$

where y is the true value, u is the upper prediction,
l is the lower prediction, and α is (100-coverage%)/100

- Winkler *"A Decision-Theoretic Approach to Interval Estimation"*, Journal of the American Statistical Association **vol 67**, pp. 187-191 (1972)
- Brehmer, Gneiting *"Scoring interval forecasts: Equal-tailed, shortest, and modal interval"*, Bernoulli **vol 27**, pp. 1993-2010 (2021)

Calculating the mean Winkler interval score in MAPIE:

```
from mapie.metrics import regression_mwi_score

regression_mwi_score(y_true, y_intervals, alpha)
```

where `y_true` are the ground truth values, and
`y_intervals` are the prediction intervals output from `mapie.predict()`

Recommended reading:

- Romano, Patterson, Candès "*Conformalized Quantile Regression*", NeurIPS **vol 32** (2019)
- Sesia, Candès "*A comparison of some conformal quantile regression methods*", Stat **vol 9** e261 (2020)
- Angelopoulos, Bates "*A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification*", arXiv:2107.07511 (2022) §2.2
- Valery Manokhin "*Practical Guide to Applied Conformal Prediction in Python*" Packt Publishing (2023) Chapter 7

Classification

Classification $\Leftarrow$ probability + decision

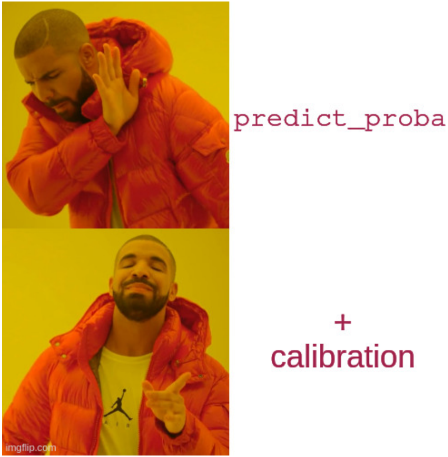
Important decisions ideally should not be taken by the ML model; they should be made by experts

(see the blog post "*Classification vs. Prediction*" by Frank E. Harrell Jr.)

`predict`

`predict_proba`

however...



For many classifiers `predict_proba` returns a relative score.

(see: Sweidan, Johansson *"Probabilistic Prediction in scikit-learn"*, diva2:1603345 (2021))

For cost-sensitive decision making, it is imperative to work with good probabilities

Motivation: ¿Which fraud case is most worth investigating first?

uncalibrated

- 500K fraud with $p = 0.61 \rightarrow$ expected cost $=$ 305K $\leftarrow$
- 1M€ fraud with $p = 0.27 \rightarrow$ expected cost $=$ 270K

calibrated

- 500K fraud with $p = 0.59 \rightarrow$ expected cost $=$ 295K
- 1M€ fraud with $p = 0.34 \rightarrow$ expected cost $=$ 340K $\leftarrow$

TL;DR:

Incorrect probabilities $\rightarrow$ incorrect decisions

Probability calibration

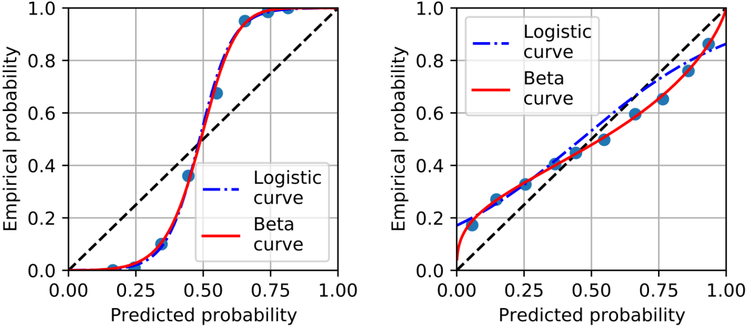
Classifier probabilities are calibrated if they are unbiased conditional on their own predictions

$$\mathbb{E}[Y|p] = p$$

Predicted probabilities are calibrated via a correction function fitted to empirical data so as to reflect the true probabilities

- Niculescu-Mizil, Caruana *"Predicting good probabilities with supervised learning"*, ICML '05, pp. 625-632 (2005)

Reliability diagram:



(a) Underconfidence (b) Overconfidence

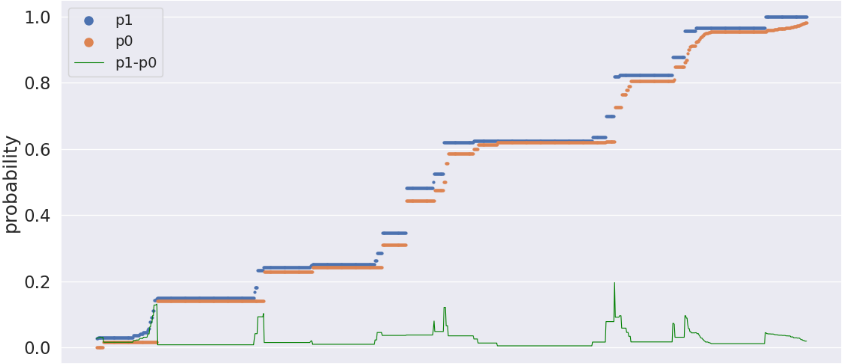
(Image from: "*Classifier calibration: a survey on how to assess and improve predicted class probabilities*")

Some probability calibration techniques

- **Platt scaling** - this is a two-parameter sigmoid correction (originally designed for the SVM classifier)
- **Isotonic scaling** - non-parametric monotonic step-wise regression fitted to class 1 for $P(1)$ values
- **Venn-ABERS** method - double isotonic; fitted to both class 1 and class 0, then merge the two via

$$p = \frac{\mathrm{fit}_{(1)}}{(1-\mathrm{fit}_{(0)})+\mathrm{fit}_{(1)}}$$

What do the fits $p1$ and $p0$ look like?



There are two implementations of Venn-ABERS predictors

- **IVAP**: inductive (`"prefit"`) - requires a hold-out calibration set
- **CVAP**: cross-conformal: fit and calibrate k times via `StratifiedKFold`

Install `venn-abers`, written by Ivan Petej

```
!pip install -q venn-abers
from venn_abers import VennAbersCalibrator
```

Example: **IVAP**: Split/"prefit" Venn-ABERS

```
p_cal  = classifier.predict_proba(X_calib)
p_test = classifier.predict_proba(X_test)

VAC = VennAbersCalibrator()
predictions = VAC.predict_proba(p_cal = p_cal,
                                y_cal = y_calib.values,
                                p_test = p_test)[:,1]
```

Example: **CVAP** cross-conformal Venn-ABERS
(no hold-out calibration set required)

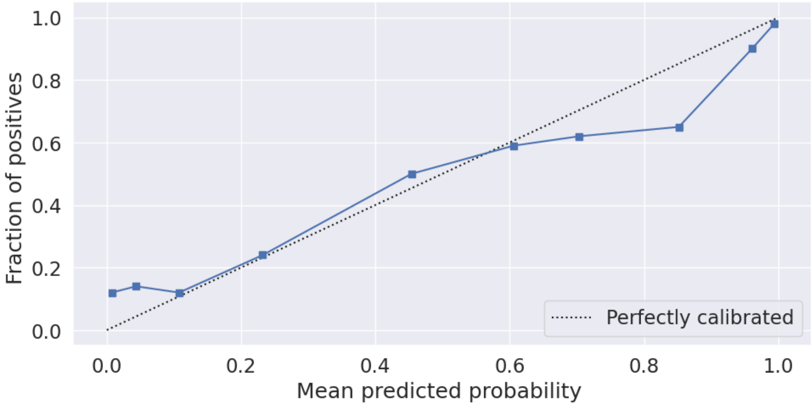
```
VAC = VennAbersCalibrator(estimator=classifier,
                          inductive=False,
                          n_splits=5)

VAC.fit(X_train, y_train)

predictions = VAC.predict_proba(X_test)[:,1]
```

Results$^{(*)}$: Base estimator XGBoost $\rightarrow$ somewhat overconfident

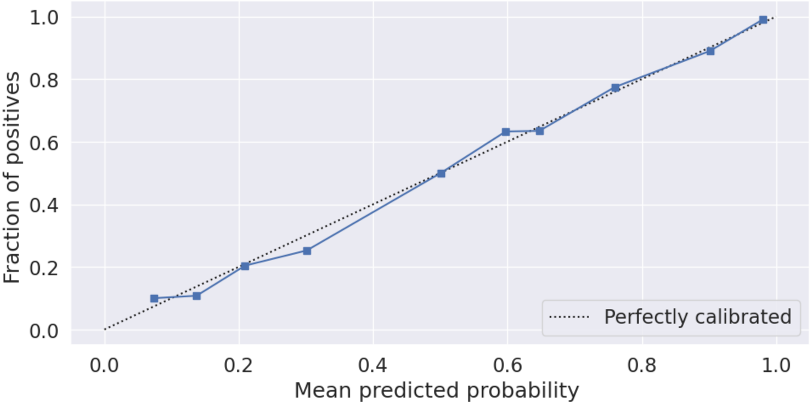
Log loss: 0.517 Brier score: 0.164



(*) Data splitting, XGBoost, and the isotonic regression contain stochastic elements, and results will vary between runs

CVAP corrected

Log loss: 0.466 Brier score: 0.153



Recommended reading:

- Vovk, Shafer, Nouretdinov "*Self-calibrating Probability Forecasting*" NIPS'03 pp. 1133-1140 (2003)
- Vovk, Petej "*Venn-Abers Predictors*" arXiv:1211.0025 (2014)
- Vovk, Petej, Fedorova "*Large-scale probabilistic prediction with and without validity guarantees*" arXiv:1511.00213 (2015)
- Valery Manokhin "*Practical Guide to Applied Conformal Prediction in Python*" Packt Publishing (2023) Chapter 6

Summary

If we want our prediction intervals to be valid,
and our classifiers to be well calibrated,
conformal prediction is the tool of choice.

Inductive CP: *¿how much calibration data should I use?*

I would suggest, both for regression and for IVAP,
to hold-out between **1000 and 2000 rows** of data.

Jupyter notebooks

Here are four notebooks where I provide working examples:

- Prediction intervals: Quantile Regression Forests
  - example of how to implement CQR "by-hand" (as per slide 19)
- Regression prediction intervals with MAPIE
  - example of using CQR via MAPIE (as per slide 22)
- Locally-weighted conformal regression
  - another powerful CP technique demonstrated "by-hand"
- Classifier calibration using Venn-ABERS
  - how to apply IVAP and CVAP (slides 36 and 37)

Some python CP libraries

- MAPIE by Thibault Cordier, Vincent Blot, Louis Lacombe,...
- Venn-ABERS calibration by Ivan Petej
- PUNCC by Mouhcine Mendil, Luca Mossina, David Vigouroux
- crepes by Henrik Boström
- AWS Fortuna by Amazon

and for much, much more a huge up-to-date collection of resources

- Awesome Conformal Prediction by Valeriy Manokhin

Books

- Valery Manokhin "Practical Guide to Applied Conformal Prediction in Python" Packt Publishing (2023)
- Christoph Molnar "Introduction To Conformal Prediction With Python" (2023)