# ML Insights Documentation

***Release 0.1.0***

**Brian Lucena and Ramesh Sampath**

**Oct 27, 2020**

# Contents

Welcome to ML_Insights, home to SplineCalib and the ModelXRay.

This package contains two main capabilities:

- SplineCalib: Spline-based probability calibration
- ModelXRay: Tool for model interpretability

CHAPTER 1

# Installation

```
$ pip install ml_insights
```

CHAPTER 2

# Probability Calibration with SplineCalib

SplineCalib is a tool for probability calibration contained in the ML-Insights package. Often, classification models may have good *discriminative* performance, but have poor *calibration*. SplineCalib post-processes the model scores so that they are better calibrated.

## 2.1 Examples

The best way to learn about SplineCalib is to work through some examples. We have provided a few below.

1. SplineCalib_Tutorial: The most basic introduction to calibration and the SplineCalib class.
2. SplineCalib_Details: A deeper dive into the various settings and parameters of the SplineCalib class.
3. SplineCalib_Multiclass_MNIST: A multiclass calibration example using the MNIST digit data.

## 2.2 SplineCalib Class

**class** ml_insights.**SplineCalib**(*penalty='l2'*, *solver='default'*, *knot_sample_size=30*, *add_knots=None*, *reg_param_vec='default'*, *cv_spline=5*, *random_state=42*, *unity_prior=False*, *unity_prior_gridsize=100*, *unity_prior_weight=100*, *max_iter=1000*, *tol=0.0001*, *logodds_scale=True*, *logodds_eps='auto'*, *reg_prec=4*, *force_knot_endpts=True*)

Probability calibration using cubic splines.

This defines a calibrator object. The calibrator can be *fit* on model outputs and truth values. After being fit, it can then be used to *calibrate* model outputs.

This is similar to the sklearn fit/transform paradigm, except that it is intended for post-processing of model outputs rather than preprocessing of model inputs.

**Parameters**

- **penalty** (*'l1' or 'l2'*) – What kind of coefficient penalization to use. The default is 'l2', which does a standard "smoothing" spline that minimizes the second derivative of the resulting function. An 'l1' penalty will typically give function which has a sparser representation in the spline bases. The 'l1' penalty can only be used with the 'liblinear' or 'saga' solver and tends to be considerably slower.
- **solver** (*'lbfgs','liblinear','newton-cg','sag','saga'*) – Which solver to use in the sklearn LogisticRegression object that powers the spline fitting. Specifying 'default' will use the 'lbfgs' for L2 penalty and 'liblinear' for L1.
- **knot_sample_size** – The number of knots to randomly sample from the training values. More knots take longer to fit. Too few knots may underfit. Too many knots could overfit, but usually the regularization will control that from happening. If *knot_sample_size* exceeds the number of unique values in the input, then all unique values will be chosen.
- **add_knots** – A list (or np_array) of knots that will be used for the spline fitting in addition to the random sample. This may be useful if you want to force certain knots to be used in areas where the data is sparse.
- **reg_param_vec** – A list (or np_array) of values to try for the 'C' parameter for regularization in the sklearn LogisticRegression. These should be positive numbers on a logarithmic scale for best results. If 'default' is chosen it will try 17 evenly spaced values (log scale) between .0001 and 10000 (inclusive)
- **cv_spline** – Number of folds to use for the cross-validation to find the best regularization parameter. Default is 5. Folds are chosen in a stratified manner.
- **random_state** – If desired, can specify the random state for the generation of the stratified folds.
- **unity_prior** – If True, routine will add synthetic data along the axis y=x as a "prior" distribution that favors that function. Default is False.
- **unity_prior_weight** – The total weight of data points added when unity_prior is set to True. Bigger values will force the calibration curve closer to the line y=x.
- **unity_prior_gridsize** – The resolution of the grid used to create the unity_prior data augmentation. Default is 100, meaning it would create synthetic data at x=0, .01 ,.02 ,. . . ,.99 ,1.
- **logodds_scale** – Whether or not to transform the x-values to the log odds scale before doing the basis expansion. Default is True and is recommended unless it is suspected that the uncalibrated probabilities already have a logistic relationship to the true probabilities.
- **logodds_eps** – Used only when logodds_scale=True. Since 0 and 1 map to positive and negative infinity on the logodds scale, we must specify a minimum and maximum probability before the transformation. Default is 'auto' which chooses a reasonable value based on the smallest positive value seen and the largest value smaller than 1.
- **reg_prec** – A positive integer designating the number of decimal places to which to round the log_loss when choosing the best regularization parameter. Algorithm breaks ties in favor of more regularization. Higher numbers will potentially use less regularization and lower numbers use more regularization. Default is 4.
- **force_knot_endpts** – If True, the smallest and largest input value will automatically chosen as knots, and *knot_sample_size*-2 knots will be chosen among the remaining values. Default is True.

**n_classes**

**Type** The number of classes for which the calibrator was fit.

**knot_vec**

**Type** The knots chosen (on the probability scale)

**knot_vec_tr**
the knots on the logodds scale. Otherwise it is the same as knot_vec.

**Type** If logodds_scale = True, this will be the values of

**lrobj**
is applied to the natural cubic spline basis. This is not used directly, but provided for reference.

**Type** (binary) The resulting sklearn LogisticRegression object that

**binary_splinecalibs**
used to do the multiclass calibration, indexed by class number.

**Type** (multiclass) A list of the binary splinecalib objects

### References

Lucena, B. Spline-Based Probability Calibration. https://arxiv.org/abs/1809.07751

**calibrate**(*y_in*)
Calibrates a set of predictions after being fit.

This function returns calibrated probabilities after being fit on a set of predictions and their true answers. It handles either binary and multiclass problems, depending on how it was fit.

**Parameters** **y_in** (*array-like, shape (n_samples, n_features)*) – The pre_calibrated scores. For binary classification can pass in a 1-d array representing the probability of class 1.

**Returns** **y_out** – The calibrated probabilities: y_out will be returned in the same shape as y_in.

**Return type** array, shape (n_samples, n_classes)

**fit**(*y_model*, *y_true*, *verbose=False*)
Fit the calibrator given a set of predictions and truth values.

This method will fit the calibrator. It handles both binary and multiclass problems.

**Parameters**

- **y_pred** (*array-like, shape (n_samples, n_classes)*) – Model outputs on which to perform calibration.

  If passed a 1-d array of length (n_samples) this will be presumed to mean binary classification and the inputs presumed to be the probability of class "1".

  If passed a 2-d array, it is assumed to be a multiclass calibration where the number of classes is n_classes. Binary problems may take 1-d or 2-d arrays as y_pred.

- **y_true** (*array-like, shape (n_samples)*) – Truth values to calibrate against. Values must be integers between 0 and n_classes-1

CHAPTER 3

# Model Interpretation with ModelXRay

ModelXRay is a tool for model interpretability contained in the ML-Insights package. It provides the capability to easily do Individual Conditional Expectation plots (ICE-plots).

CHAPTER 4

# Indices

- genindex

# Index