py-ciu

Kary Främling

Dec 05, 2023

CONTENTS:

This package implements the Contextual Importance and Utility (CIU) method.

Classes:

- *ciu.CIU*: The CIU class implements the Contextual Importance and Utility method for Explainable AI.
- *ciu.PerturbationMinMaxEstimator.PerturbationMinMaxEstimator*: Class the finds minimal and maximal output values by perturbation of input value(s). This is the default class/method used by *CIU*.

Functions:

- *ciu.CIU.contrastive_ciu*: Function for calculating contrastive values from two CIU results.

Example:

```
# Example code using the module
import ciu as ciu
CIU = ciu.CIU(model.predict_proba, ['Output Name(s)'], data=X_train)
CIUres = CIU.explain(instance)
print(CIUres)
```

class ciu.CIU.**CIU**(*predictor*, *out_names*, *data=None*, *input_names=None*, *in_minmaxs=None*, *out_minmaxs=None*, *nsamples=100*, *category_mapping=None*, *neutralCU=0.5*, *output_inds=[0]*, *vocabulary=None*, *minmax_estimator=None*)

The CIU class implements the Contextual Importance and Utility method for Explainable AI.

The method *explain_core()* contains all the CIU mathematics. However, it is probably not the method that would normally be called directly because it estimates CIU for a coalition of inputs (which works both for individual features and for CIU's *Intermediate Concepts*). It returns a list of DataFrames with CIU results, where each DataFrame corresponds to the explanation of one output.

The methods that would normally be called are *explain()* (for individual features), *explain_voc()* for Intermediate Concepts (/coalitions of features), and *explain_all()* for a set of instances. These all return a DataFrame with (presumably) all useful CIU result information (CI, CU, Contextual influence etc.).

Then there are also various methods for presenting CIU results graphically and textually. Some of these wouldn't necessarily have to be methods of the *CIU* class but they have been included here as a compromise.

Parameters

- **predictor** – Model prediction function to be used.
- **out_names** (*[str]*) – List of names for the model outputs. This parameter is compulsory because it is used for determining how many outputs there are and initializing `out_minmaxs` to 0/1 if they are not provided as parameters.
- **data** (*DataFrame*) – Data set to use for inferring min and max input values. Only needed if `in_minmaxs` is not provided.
- **input_names** (*[str]*) – list of input column names in `data`.
- **in_minmaxs** (*DataFrame*) – Pandas DataFrame with columns `min` and `max` and one row per input. If this parameter is provided, then `data` does not need to be passed.
- **out_minmaxs** (*DataFrame*) – Pandas DataFrame with columns `min` and `max` and one row per model output. If the value is `None`, then `out_minmaxs` is initialized to `[0,1]` for all outputs. In practice this signifies that this parameter is typically not needed for classification tasks but is necessary to provide or regression tasks.
- **nsamples** (*int*) – Number of samples to use for estimating CIU of numerical inputs.

- **category_mapping** (*dict*) – Dictionary that contains names of features that should be dealt with as categories, i.e. having discrete int/str values. The use of this mapping is strongly recommended for efficiency and accuracy reasons! In the "R" implementation such a mapping is not needed because the *factor* column type indicates the columns and the possible values. The corresponding *Categorical* type doesn't seem to be used consistently in Python ML libraries so it didn't seem like a good choice to use that for the moment.
- **neutralCU** (*float*) – Reference/baseline value to use for Contextual influence.
- **output_inds** (*[int]*) – Default output index/indices to explain. This value doesn't have to be given as a list, it can also be a single integer (that is automatically converted into a list).
- **vocabulary** (*DataFrame*) – Vocabulary to use, defined as a DataFrame.
- **minmax_estimator** (*object*) – Class to be used for estimating ymin/ymax values, if something else is to be used than the default one.

explain(*instance=None*, *output_inds=None*, *input_inds=None*, *nsamples=None*, *neutralCU=None*, *vocabulary=None*, *target_concept=None*, *target_ciu=None*)

Determines contextual importance and utility for a given instance (set of input/feature values). This method calculates CIU values only for individual features (not for Intermediate Concepts / coalitions of features), so if `input_inds` is given, then the returned CIU DataFrame will have the individual CI, CU etc values. If `input_inds=None`, then CIU results are returned for all inputs/features.

Parameters

- **instance** (*DataFrame*) – Instance to be explained. If `instance=None` then the last passed instance is used by default.
- **output_inds** (*[int]*) – Index of model output to explain. Default is None, in which case it is the `output_inds` value given to the *CIU* constructor. This value doesn't have to be given as a list, it can also be a single integer (that is automatically converted into a list).
- **input_inds** (*[int]*) – List of input indices to include in explanation. Default is None, which signifies "all inputs".
- **nsamples** (*int*) – Number of samples to use. Default is `None`, which means using the value of the *CIU* constructor.
- **neutralCU** (*int*) – Value to use for "neutral CU" in Contextual influence calculation. Default is `None` because this parameter is only intended to temporarily override the value given to the *CIU* constructor.
- **vocabulary** (*DataFrame*) – Vocabulary to use, defined as a DataFrame. Only needed for overriding the default vocabulary given to *CIU* constructor and if there's a `target_concept`.
- **target_concept** (*str*) – Name of target concept, if the explanation is for an intermediate concept rather than for the output value.
- **target_ciu** (*DataFrame*) – If a CIU result already exists for the target_concept, then it can be passed with this parameter. Doing so avoids extra calculations and also avoids potential noise due to perturbation randomness in CIU calculations.

Returns

DataFrame with CIU results for the requested output(s).

explain_all(*data=None*, *output_inds=None*, *input_inds=None*, *nsamples=None*, *neutralCU=None*, *vocabulary=None*, *target_concept=None*, *target_ciu=None*, *do_norm_invals=False*)

Do CIU for all instances in *data*.

Parameters

- **data** (*DataFrame*) – DataFrame with all instances to evaluate.
- **output_inds** (*[int]*) – See *explain()*.
- **input_inds** (*[int]*) – See *explain()*.
- **nsamples** (*int*) – See *explain()*.
- **neutralCU** (*int*) – See *explain()*.
- **vocabulary** (*DataFrame*) – See *explain()*.
- **target_concept** (*str*) – See *explain()*.
- **target_ciu** (*DataFrame*) – See *explain()*.

Param
do_norm_invals: Should a column with normalized input values be produced or not? This can only be done for "basic" features, not for coalitions of features (intermediate concepts) at least for the moment. It is useful to provide normalized input values for getting more meaningful beeswarm plots, for instance.

Returns
DataFrame with CIU results of all instances concatenated.

explain_core(*coalition_inputs*, *instance=None*, *output_inds=None*, *feature_name=None*, *nsamples=None*, *neutralCU=None*, *target_inputs=None*, *out_minmaxs=None*, *target_concept=None*)

Calculate CIU for a coalition of inputs. This is the "core" CIU method with the actual CIU calculations. All other methods should call this one for doing actual CIU calculations.

Coalitions of inputs are used for defining CIU's "intermediate concepts". It signifies that all the inputs in the coalition are perturbed at the same time.

Parameters

- **coalition_inputs** (*[int]*) – list of input indices.
- **instance** (*DataFrame*) – Instance to be explained. If `instance=None` then the last passed instance is used by default.
- **output_inds** – See corresponding parameter of *CIU* constructor method. Default value `None` will use the value given to constructor method.
- **feature_name** (*str*) – Feature name to use for coalition of inputs (i.e. if more than one input index is given), instead of the default "Coalition of…" feature name.
- **nsamples** – See corresponding parameter of constructor method. Default value `None` will use the value given to constructor method.
- **neutralCU** – See corresponding parameter of constructor method. Default value `None` will use the value given to constructor method.
- **target_inputs** (*[int]*) – list of input indices for "target" concept, i.e. a CIU "intermediate concept". Normally "coalition_inputs" should be a subset of "target_inputs" but that is not a requirement, mathematically taken. Default is None, which signifies that the model outputs (i.e. "all inputs") are the targets and the "out_minmaxs" values are used for CI calculation.
- **out_minmaxs** (*DataFrame*) – DataFrame with min/max output values to use instead of the "global" ones. This is used for implementing Intermediate Concept calculations. The DataFrame must have one row per output and two columns, preferably named *ymin* and *ymax*.

- **target_concept** (*str*) – Name of the target concept. This is not used for calculations, it is only for filling up the `target_concept` coliumn of the CIU results.

Returns

A `list` of DataFrames with CIU results, one for each output of the model. **Remark:** *explain_core()* indeed returns a *list*, which is a difference compared to the two other *explain_* methods!

explain_voc(*instance=None*, *output_inds=None*, *input_concepts=None*, *nsamples=None*, *neutralCU=None*, *vocabulary=None*, *target_concept=None*, *target_ciu=None*)

Determines contextual importance and utility for a given instance (set of input/feature values), using the intermediate concept vocabulary.

Parameters

- **instance** (*DataFrame*) – See *explain()*.
- **output_inds** (*[int]*) – See *explain()*.
- **input_concepts** (*[str]*) – List of concepts to include in the explanation. Default is None, which signifies "all concepts in the vocabulary".
- **nsamples** (*int*) – See *explain()*.
- **neutralCU** (*int*) – See *explain()*.
- **vocabulary** (*DataFrame*) – Vocabulary to use, defined as a DataFrame. Only needed for overriding the default vocabulary given to *CIU* constructor.
- **target_concept** (*str*) – See *explain()*.
- **target_ciu** (*DataFrame*) – See *explain()*.

Returns

DataFrame with CIU results for the requested output(s).

plot_3D(*ind_inputs*, *instance=None*, *ind_output=0*, *nbr_pts=(40, 40)*, *figsize=(6, 6)*, *azim=None*)

Plot output value as a function of two inputs.

Parameters

- **ind_inputs** (*list*) – indexes for two features to use for the 3D plot.
- **instance** (*pd.DataFrame*) – instance to use if it is different from given to the constructor or as a parameter to `explain()``or `èxplain_voc()`. Default: None.
- **ind_output** (*int*) – index of output to plot. Default: 0.
- **nbr_pts** (*int*) – number of points to use (both axis). Default: (40,40).
- **figsize** – Values to pass to `plt.figure()`. Default: (6,6).
- **azim** – azimuth angle to use. Default: None.

Returns

3D plot object

plot_ciu(*ciu_result=None*, *plot_mode='default'*, *CImax=1.0*, *sort='CI'*, *main=None*, *color_blind=None*, *figsize=(6, 4)*, *color_fill_ci='#7fffd44d'*, *color_edge_ci='#66CDAA'*, *color_fill_cu='#006400cc'*, *color_edge_cu='#006400'*)

The core plotting method for CIU results, which uses both CI and CU values in the explanation.

Parameters

- **ciu_result** (*DataFrame*) – CIU result DataFrame as returned by one of the "explain…" methods.
- **plot_mode** (*str*) – defines the type plot to use between 'default', 'overlap' and 'combined'.
- **CImax** – Limit CI axis to the given value. Default is 1
- **sort** (*str*) – defines the order of the plot bars by the 'CI' (default), 'CU' values or unsorted if None;
- **color_blind** (*str*) – defines accessible color maps to use for the plots, such as 'protanopia', 'deuteranopia' and 'tritanopia'.
- **color_edge_cu** (*str*) – defines the hex or named color for the CU edge in the overlap plot mode.
- **color_fill_cu** (*str*) – defines the hex or named color for the CU fill in the overlap plot mode.
- **color_edge_ci** (*str*) – defines the hex or named color for the CI edge in the overlap plot mode.
- **color_fill_ci** (*str*) – defines the hex or named color for the CI fill in the overlap plot mode.

plot_influence(*ciu_result*, *xminmax=None*, *main=None*, *figsize=(6, 4)*, *colors=('firebrick', 'steelblue')*, *edgecolors=('#808080', '#808080')*)

Plot CIU result as a bar plot using Contextual influence values.

Parameters

- **ciu_result** – CIU result DataFrame as returned by one of the "explain…" methods.
- **xminmax** – Range to pass to `xlim`. Default: None.
- **figsize** – Value to pass as `figsize` parameter. Default: (6, 6).
- **colors** – Bar colors to use. Default: ("firebrick","steelblue").
- **edgecolors** – Bar edge colors to use. Default: ("firebrick","steelblue").

plot_input_output(*instance=None*, *ind_input=0*, *output_inds=0*, *in_min_max_limits=None*, *n_points=40*, *main=None*, *xlab='x'*, *ylab='y'*, *ylim=0*, *figsize=(6, 4)*, *illustrate_CIU=False*, *legend_location=0*, *neutral_CU=0.5*, *CIU_illustration_colours=('red', 'green')*)

Plot model output(s) value(s) as a function on one input. Works both for numerical and for categorical inputs.

Parameters

- **instance** (*DataFrame*) – See *explain()*. If *None*, then use last instance passed to an *explain_()* method.
- **ind_input** (*int*) – Index of input to use.
- **output_inds** (*int, [int], None*) – Integer value, list of integers or None. If None then all outputs are plotted. Default: 0.
- **in_min_max_limits** (*int array/list*) – Limits to use for input values. If None, the default ones are used.
- **n_points** (*int*) – Number of x-values to use for numerical inputs.
- **xlab** (*str*) – X-axis label.
- **ylab** (*str*) – Y-axis label.

- **ylim** (*int, (min, max), None*) – Value limits for y-axis. Can be zero, actual limits or None. Zero signifies that the known min/max values for the output will be used. `None` signifies that no limits are defined and are auto-determined by `plt.plot`. If actual limits are given, they are passed to `plt.ylim` as such. Default: zero.
- **figsize** (*(int,int)*) – Figure size to use.
- **illustrate_CIU** (*boolean*) – Plot CIU illustration or not?
- **legend_location** – See `matplotlib.pyplot.legend()`
- **neutral_CU** (*float*) – Neutral CU value to use for plotting Contextual influence reference value.
- **CIU_illustration_colours** – Colors to use for CIU illustration, in order: (*ymin*,`ymax`)

textual_explanation(*ciu_result*, *target_ciu=None*, *thresholds_ci=None*, *thresholds_cu=None*, *use_markdown_effects=False*)

Parameters

- **thresholds_ci** (*dict*) – dictionary containing the label and ceiling value for the CI thresholds
- **thresholds_cu** (*dict*) – dictionary containing the label and ceiling value for the CU thresholds

Returns
Explanation as *str*.

ciu.CIU.**contrastive_ciu**(*ciures1*, *ciures2*)

Calculate contrastive influence values for two CIU result DataFrames.

The two DataFrames should have the same features, in the same order.

Parameters

- **ciures1** (*DataFrame*) – CIU result DataFrame of the "focus" instance.
- **ciures2** (*DataFrame*) – CIU result DataFrame of the "challenger" instance.

Returns
list with one influence value per feature/concept.

class ciu.PerturbationMinMaxEstimator.**PerturbationMinMaxEstimator**(*predictor*, *in_minmaxs*, *nsamples*)

This class is for abstracting the operation of finding minimal and maximal output value(s) for a given instance and given inputs (input indices).

PerturbationMinMaxEstimator is mainly meant to be used from CIU, not directly! It is the default claas used by CIU for finding minimal and maximal output values but it can be replaced by some other class/object that does it in some (presumably) more efficient way. This can be useful if some model-specific knowledge is available or if there's a reason to do the sampling in a more in-distribution way.

The only compulsory method is `get_minmax_outvals`, which is the method called by CIU with the parameters *ìnstance* ` and `indices`.

Parameters

- **predictor** – The predictor function to call.
- **in_minmaxs** – DataFrame with as many rows as features and two columns with min and max feature values, respectively.

- **nsamples** (*int*) – How many samples to use.

`get_minmax_outvals`(*instance*, *indices*, *category_mapping=None*)

Find the minimal and maximal output value(s) that can be obtained by modifying the inputs `indices` of the instance `instance`.

Parameters

- **instance** – The instance to generate the permuted instances for.
- **indices** – list of indices for which to generate perturbed values.

Returns

Two np.arrays with mininmal and maximal output values found for the input or coalition of inputs in `indices`.

`ciu.ciuplots.`**ciu_beeswarm**(*df*, *xcol='CI'*, *ycol='feature'*, *color_col='norm_invals'*, *legend_title=None*, *jitter_level=0.5*, *palette=['blue', 'red']*, *opacity=0.8*)

Create a beeswarm plot of values. This can be used for CI, Cinfl, CU or any values in principle (including Shapley value, LIME values, ...).

Remark: This has not been tested/implemented for non-numerical values, intermediate concepts etc. (unlike the R version).

Param

df: A "long" CIU result DataFrame, typically produced by a call to *ciu.CIU.CIU.explain_all()*.

Parameters

- **xcol** (*str*) – Name of column to use for X-axis (numerical).
- **ycol** (*str*) – Name of column to use for Y-axis, typically the one that contains feature names.
- **color_col** (*str*) – Name of column to use for dot color, typically the one that instance/feature values that are normalised into *[0,1]* interval.
- **legend_title** (*str*) – Text to use as legend title. If *None*, then used *color_col*.
- **jitter_level** (*float*) – Level of jitter to use.
- **palette** (*list*) – Color palette to use. The default value is a list with two colors but can probably be any kind of palette that is accepted by plotly.graphobjects.
- **opacity** (*float*) – Opacity value to use for dots.

Returns

A plotly.graphobjects Figure.

`ciu.ciuplots.`**plot_contrastive**(*ciures1*, *ciures2*, *xminmax=None*, *main=None*, *figsize=(6, 4)*, *colors=('firebrick', 'steelblue')*, *edgecolors=('#808080', '#808080')*)

Create a contrastive plot for the two CIU results passed. This is essentially similar to an influence plot.

Parameters

- **ciures1** (*DataFrame*) – See *ciu.CIU.contrastive_ciu()*
- **ciures2** (*DataFrame*) – See *ciu.CIU.contrastive_ciu()*
- **xminmax** (*array/list*) – Min/max values to use for X axis.
- **main** (*str*) – Main title to use.
- **figsize** (*array*) – Figure size.
- **colors** (*array*) – Bar colors to use.

- **edgecolors** (*array*) – Bar edge colors to use.

:return A pyplot plot.

CHAPTER

ONE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

C

INDEX