causeinfer Documentation

Release 1.0.1

andrewtavis

Jun 03, 2022

CONTENTS:

Machine learning based causal inference/uplift in Python

```
pip install causeinfer
```

```
git clone https://github.com/andrewtavis/causeinfer.git
cd causeinfer
python setup.py install
```

```
import causeinfer
```

CHAPTER

ONE

STANDARD_ALGORITHMS

The `standard_algorithms` module compiles causal inference modeling techniques for quick application.

1.1 Base Models

Base models for the following algorithms:

- The Two Model Approach
- The Interaction Term Approach
- The Binary Class Transformation (BCT) Approach
- The Quaternary Class Transformation (QCT) Approach
- The Reflective Uplift Approach
- The Pessimistic Uplift Approach

Note: these classes should not be used directly. Please use derived classes instead.

Based on

Kuchumov, A. pyuplift: Lightweight uplift modeling framework for Python. (2019). URL: https://github.com/duketemon/pyuplift. License: https://github.com/duketemon/pyuplift/blob/master/LICENSE.

Contents

BaseModel Class
fit, predict

TransformationModel Class (see annotation/methodology explanation)
is_treatment_positive, is_control_positive, is_control_negative, is_treatment_negative

`class causeinfer.standard_algorithms.base_models.BaseModel`

Base class for the Two Model and Interaction Term Approaches.

`fit`(*X*, *y*, *w*)

Parameters

X
[numpy.ndarray][(num_units, num_features)][int, float] Dataframe of covariates.

y
[numpy.ndarray][(num_units,)][int, float] Vector of unit responses.

w
[numpy.ndarray][(num_units,)][int, float] Designates the original treatment allocation across units.

Returns

self
[object]

`predict`(*X*, *w*)

Parameters

X
[numpy.ndarray][(num_pred_units, num_pred_features)][int, float] New data on which to make a prediction.

w
[numpy.ndarray][(num_pred_units, num_pred_features)][int, float] Treatment allocation for predicted units.

Returns

y_pred
[numpy.ndarray][(num_pred_units,)][int, float] Vector of predicted unit responses.

`class causeinfer.standard_algorithms.base_models.TransformationModel`

Base class for the Response Transformation Approaches.

Notes

The following is non-standard annotation to combine marketing and other methodologies.

Traditional marketing annotation is found in parentheses.

The response transformation approach splits the units based on response and treatment:

- TP : Treatment Positives (Treatment Responders).
- CP : Control Positives (Control Responders).
- CN : Control Negatives (Control Nonresponders).
- TN : Treatment Negatives (Treatment Nonresponders).

From these four known classes we want to derive the characteristic responses of four unknown classes:

- AP : Affected Positives (Persuadables) : within TPs and CNs.
- UP : Unaffected Positives (Sure Things) : within TPs and CPs.
- UN : Unaffected Negatives (Lost Causes) : within CNs and TNs.
- AN : Affected Negatives (Do Not Disturbs) : within CPs and TNs.

The focus then falls onto predicting APs and ANs via their known classes.

`is_treatment_positive`(*y*, *w*)

Checks if a subject did respond when treated.

Parameters

y
[int, float] The target response.

w
[int, float] The treatment value.

Returns

is_treatment_positive
[bool]

`is_control_positive`(*y*, *w*)

Checks if a subject did respond when not treated.

Parameters

y
[int, float] The target response.

w
[int, float] The treatment value.

Returns

is_control_positive
[bool]

`is_control_negative`(*y*, *w*)

Checks if a subject didn't respond when not treated.

Parameters

y
[int, float] The target response.

w
[int, float] The treatment value.

Returns

is_control_negative
[bool]

`is_treatment_negative`(*y*, *w*)

Checks if a subject didn't respond when treated.

Parameters

y
[int, float] The target response.

w
[int, float] The treatment value.

Returns

is_treatment_negative
[bool]

1.2 Two Model

The Two Model Approach (Double Model, Separate Model).

Based on

Kuchumov, A. pyuplift: Lightweight uplift modeling framework for Python. (2019). URL: https://github.com/duketemon/pyuplift. License: https://github.com/duketemon/pyuplift/blob/master/LICENSE.

Hansotia, B. and B. Rukstales (2002). "Incremental value modeling". In: Journal of Interactive Marketing 16(3), pp. 35–46. URL: https://search.proquest.com/openview/1f86b52432f7d80e46101b2b4b7629c0/1?cbl=32002&pq-origsite=gscholar

Devriendt, F. et al. (2018). A Literature Survey and Experimental Evaluation of the State-of-the-Art in Uplift Modeling: A Stepping Stone Toward the Development of Prescriptive Analytics. Big Data, Vol. 6, No. 1, March 1, 2018, pp. 1-29. Codes found at: data-lab.be/downloads.php.

Contents

TwoModel Class
fit, predict, predict_proba

`class causeinfer.standard_algorithms.two_model.TwoModel(control_model=None, treatment_model=None)`

`fit(X, y, w)`

Trains a model given covariates, responses and assignments.

Parameters

X
[numpy.ndarray][(num_units, num_features)][int, float] Matrix of covariates.

y
[numpy.ndarray][(num_units,)][int, float] Vector of unit responses.

w
[numpy.ndarray][(num_units,)][int, float] Vector of original treatment allocations across units.

Returns

treatment_model, control_model
[causeinfer.standard_algorithms.TwoModel] Two trained models (one for training group, one for control).

`predict(X)`

Predicts a causal effect given covariates.

Parameters

X
[numpy.ndarray][(num_units, num_features)][int, float] New data on which to make predictions.

Returns

predictions
[numpy.ndarray][(num_units, 2)][float] Predicted causal effects for all units given treatment model and control.

`predict_proba`(*X*)

Predicts the probability that a subject will be a given class given covariates.

Parameters

X
[numpy.ndarray][(num_units, num_features)][int, float] New data on which to make predictions.

Returns

probas
[numpy.ndarray][(num_units, 2)][float] Predicted probability to respond for all units given treatment and control models.

1.3 Interaction Term

The Interaction Term Approach (The True Lift Model, The Dummy Variable Approach).

Based on

Kuchumov, A. pyuplift: Lightweight uplift modeling framework for Python. (2019). URL: https://github.com/duketemon/pyuplift. License: https://github.com/duketemon/pyuplift/blob/master/LICENSE.

Lo, VSY. (2002). "The true lift model: a novel data mining approach to response modeling in database marketing". In:SIGKDD Explor4 (2), 78–86. URL: https://dl.acm.org/citation.cfm?id=772872

Devriendt, F. et al. (2018). A Literature Survey and Experimental Evaluation of the State-of-the-Art in Uplift Modeling: A Stepping Stone Toward the Development of Prescriptive Analytics. Big Data, Vol. 6, No. 1, March 1, 2018, pp. 1-29. Codes found at: data-lab.be/downloads.php.

Contents

InteractionTerm Class
fit, predict, predict_proba

`class causeinfer.standard_algorithms.interaction_term.InteractionTerm`(*model=None*)

`fit`(*X*, *y*, *w*)

Trains a model given covariates, responses and assignments.

Parameters

X
[numpy.ndarray][(num_units, num_features)][int, float] Matrix of covariates.

y
[numpy.ndarray][(num_units,)][int, float] Vector of unit responses.

w
[numpy.ndarray][(num_units,)][int, float] Vector of original treatment allocations across units.

Returns

self
[causeinfer.standard_algorithms.InteractionTerm] A trained model.

`predict`(*X*)

Predicts a causal effect given covariates.

Parameters

X
[numpy.ndarray][(num_units, num_features)][int, float] New data on which to make predictions.

Returns

predictions
[numpy.ndarray][(num_units, 2)][float] Predicted causal effects for all units given a 1 and 0 interaction term.

`predict_proba`(*X*)

Predicts the probability that a subject will be a given class given covariates.

Parameters

X
[numpy.ndarray][(num_units, num_features)][int, float] New data on which to make predictions.

Returns

probas
[numpy.ndarray][(num_units, 2)][float] Predicted causal probabilities for all units given a 1 and 0 interaction term.

1.4 Binary Class Transformation

The Binary Class Transformation Approach (Influential Marketing, Response Transformation Approach).

Based on

Kuchumov, A. pyuplift: Lightweight uplift modeling framework for Python. (2019). URL: https://github.com/duketemon/pyuplift. License: https://github.com/duketemon/pyuplift/blob/master/LICENSE.

Lai, L.Y.-T. (2006). "Influential marketing: A new direct marketing strategy addressing the existence of voluntary buyers". Master of Science thesis, Simon Fraser University School of Computing Science, Burnaby, BC,Canada. URL: https://summit.sfu.ca/item/6629

Shaar, A., Abdessalem, T., and Segard, O. (2016). "Pessimistic Uplift Modeling". ACM SIGKDD, August 2016, San Francisco, California USA, arXiv:1603.09738v1. URL:https://pdfs.semanticscholar.org/a67e/401715014c7a9d6a6679df70175be01daf7c.pdf.

Devriendt, F. et al. (2018). A Literature Survey and Experimental Evaluation of the State-of-the-Art in Uplift Modeling: A Stepping Stone Toward the Development of Prescriptive Analytics. Big Data, Vol. 6, No. 1, March 1, 2018, pp. 1-29. Codes found at: data-lab.be/downloads.php.

Contents

BinaryTransformation Class
_binary_transformation, _binary_regularization, fit, predict (Not available at this time), predict_proba

`class causeinfer.standard_algorithms.binary_transformation.BinaryTransformation`(*model=None*, *regularize=False*)

`_binary_transformation`(*y*, *w*)

Derives which of the unknown Affected Positive or Affected Negative classes the unit could fall into based known outcomes.

Parameters

y
[numpy.ndarray][(num_units,)][int, float] Vector of unit responses.

w
[numpy.ndarray][(num_units,)][int, float] Vector of original treatment allocations across units.

Returns

np.array(y_transformed)
[numpy.ndarray][an array of transformed unit classes.]

`_binary_regularization`(*y=None*, *w=None*)

Regularization of binary classes is based on the positive and negative binary affectual classes.

Parameters

y
[numpy.ndarray][(num_units,)][int, float] Vector of unit responses.

w
[numpy.ndarray][(num_units,)][int, float] Vector of original treatment allocations across units.

Returns

fav_ratio, unfav_ratio
[float] Regularized ratios of favorable and unfavorable classes.

`fit`(*X*, *y*, *w*)

Trains a model given covariates, responses and assignments.

Parameters

X
[numpy.ndarray][(num_units, num_features)][int, float] Matrix of covariates.

y
[numpy.ndarray][(num_units,)][int, float] Vector of unit responses.

w
[numpy.ndarray][(num_units,)][int, float] Vector of original treatment allocations across units.

Returns

self
[causeinfer.standard_algorithms.BinaryTransformation] A trained model.

`predict_proba`(*X*)

Predicts the probability that a subject will be a given class given covariates.

Parameters

X
[numpy.ndarray][(num_units, num_features)][int, float] New data on which to make predictions.

Returns

probas
[numpy.ndarray][(num_units, 2)][float] Predicted probabilities for being a favorable class and unfavorable class.

1.5 Quaternary Class Transformation

The Quaternary Class Transformation Approach (Response Transformation Approach).

Based on

Kuchumov, A. pyuplift: Lightweight uplift modeling framework for Python. (2019). URL: https://github.com/duketemon/pyuplift. License: https://github.com/duketemon/pyuplift/blob/master/LICENSE.

Kane, K., Lo, VSY., and Zheng, J. (2014). "Mining for the truly responsive customers and prospects using truelift modeling: Comparison of new and existing methods". In:Journal of Marketing Analytics 2(4), 218–238. URL: https://link.springer.com/article/10.1057/jma.2014.18

Devriendt, F. et al. (2018). A Literature Survey and Experimental Evaluation of the State-of-the-Art in Uplift Modeling: A Stepping Stone Toward the Development of Prescriptive Analytics. Big Data, Vol. 6, No. 1, March 1, 2018, pp. 1-29. Codes found at: data-lab.be/downloads.php.

Contents

QuaternaryTransformation Class

_quaternary_transformation, _quaternary_regularization, fit, predict (not available at this time), predict_proba

class causeinfer.standard_algorithms.quaternary_transformation.**QuaternaryTransformation**(*model=None*, *regularize=False*)

_quaternary_transformation(*y*, *w*)

Assigns known quaternary (TP, CP, CN, TN) classes to units.

Parameters

y

[numpy.ndarray][(num_units,)][int, float] Vector of unit responses.

w

[numpy.ndarray][(num_units,)][int, float] Vector of original treatment allocations across units.

Returns

np.array(y_transformed)

[np.array] an array of transformed unit classes.

_quaternary_regularization(*y=None*, *w=None*)

Regularization of quaternary classes is based on their treatment assignment.

Parameters

y

[numpy.ndarray][(num_units,)][int, float] Vector of unit responses.

w

[numpy.ndarray][(num_units,)][int, float] Vector of original treatment allocations across units.

Returns

control_count, treatment_count

[int] Regularized amounts of control and treatment classes.

`fit`(*X*, *y*, *w*)

Trains a model given covariates, responses and assignments.

Parameters

X
[numpy.ndarray][(num_units, num_features)][int, float] Matrix of covariates.

y
[numpy.ndarray][(num_units,)][int, float] Vector of unit responses.

w
[numpy.ndarray][(num_units,)][int, float] Vector of original treatment allocations across units.

Returns

self
[causeinfer.standard_algorithms.QuaternaryTransformation] A trained model.

`predict_proba`(*X*)

Predicts the probability that a subject will be a given class given covariates.

Parameters

X
[numpy.ndarray][(num_units, num_features)][int, float] New data on which to make predictions.

Returns

probas
[numpy.ndarray][(num_units, 2)][float] Predicted probabilities for being a favorable class and an unfavorable class.

1.6 Reflective Uplift Transformation

The Reflective Uplift Transformation Approach.

Based on

Kuchumov, A. pyuplift: Lightweight uplift modeling framework for Python. (2019). URL: https://github.com/duketemon/pyuplift. License: https://github.com/duketemon/pyuplift/blob/master/LICENSE.

Shaar, A., Abdessalem, T., and Segard, O. (2016). "Pessimistic Uplift Modeling". ACM SIGKDD, August 2016, San Francisco, California USA, arXiv:1603.09738v1. URL:https://pdfs.semanticscholar.org/a67e/401715014c7a9d6a6679df70175be01daf7c.pdf.

Contents

ReflectiveUplift Class
fit, predict (not available at this time), predict_proba, _reflective_transformation, _reflective_weights

`class causeinfer.standard_algorithms.reflective.`**`ReflectiveUplift`**(*model=None*)

`fit`(*X*, *y*, *w*)

Trains a model given covariates, responses and assignments.

Parameters

X
[numpy.ndarray][(num_units, num_features)][int, float] Matrix of covariates.

y
[numpy.ndarray][(num_units,)][int, float] Vector of unit responses.

w
[numpy.ndarray][(num_units,)][int, float] Vector of original treatment allocations across units.

Returns

self
[causeinfer.standard_algorithms.ReflectiveUplift] A trained model.

`predict_proba`(*X*)

Predicts the probability that a subject will be a given class given covariates.

Parameters

X
[numpy.ndarray][(num_units, num_features)][int, float] New data on which to make predictions.

Returns

probas
[numpy.ndarray][(num_units, 2)][float] Predicted probabilities for being a favorable class and an unfavorable class.

`_reflective_transformation`(*y*, *w*)

Assigns known quaternary (TP, CP, CN, TN) classes to units.

Parameters

y
[numpy.ndarray][(num_units,)][int, float] Vector of unit responses.

w
[numpy.ndarray][(num_units,)][int, float] Vector of original treatment allocations across units.

Returns

np.array(y_transformed)
[np.array] an array of transformed unit classes.

`_reflective_weights`(*y*, *w*)

Derives weights to normalize binary transformation noise.

Parameters

y
[numpy.ndarray][(num_units,)][int, float] Vector of unit responses.

w
[numpy.ndarray][(num_units,)][int, float] Vector of original treatment allocations across units.

Returns

p_tp_fav, p_cp_fav, p_cn_unfav, p_tn_unfav
[np.array] Probabilities of being a quaternary class per binary class.

1.7 Pessimistic Uplift Transformation

The Pessimistic Uplift Transformation Approach.

Based on

Kuchumov, A. pyuplift: Lightweight uplift modeling framework for Python. (2019). URL: https://github.com/duketemon/pyuplift. License: https://github.com/duketemon/pyuplift/blob/master/LICENSE.

Shaar, A., Abdessalem, T., and Segard, O. (2016). "Pessimistic Uplift Modeling". ACM SIGKDD, August 2016, San Francisco, California USA, arXiv:1603.09738v1. URL:https://pdfs.semanticscholar.org/a67e/401715014c7a9d6a6679df70175be01daf7c.pdf.

Contents

PessimisticUplift Class

fit, predict (not available at this time), predict_proba

`class causeinfer.standard_algorithms.pessimistic.PessimisticUplift(model=None)`

`fit(X, y, w)`

Trains a model given covariates, responses and assignments.

Parameters

X

[numpy.ndarray][(num_units, num_features)][int, float] Matrix of covariates.

y

[numpy.ndarray][(num_units,)][int, float] Vector of unit responses.

w

[numpy.ndarray][(num_units,)][int, float] Vector of original treatment allocations across units.

Returns

self

[causeinfer.standard_algorithms.PessimisticUplift] A trained model.

`predict_proba(X)`

Predicts the probability that a subject will be a given class given covariates.

Parameters

X

[numpy.ndarray][(num_units, num_features)][int, float] New data on which to make predictions.

Returns

probas

[numpy.ndarray][(num_units, 2)][float] Predicted probabilities for being a favorable class and an unfavorable class.

CHAPTER

TWO

EVALUATION

The `evaluation` module provides methods for accuracy measurement and presentation.

2.1 Metrics

causeinfer metrics provide statistical impressions of model performance.

Functions

- *causeinfer.evaluation.get_cum_effect()*
- *causeinfer.evaluation.get_cum_gain()*
- *causeinfer.evaluation.get_qini()*
- *causeinfer.evaluation.auuc_score()*
- *causeinfer.evaluation.qini_score()*
- causeinfer.evaluation.get_batch_metrics()
- *causeinfer.evaluation.signal_to_noise()*

causeinfer.evaluation.**get_cum_effect**(*df*, *models=None*, *outcome_col='y'*, *treatment_col='w'*, *treatment_effect_col='tau'*, *normalize=False*, *random_seed=None*)

Gets average causal effects of model estimates in cumulative population.

Parameters

df
[pandas.DataFrame] A data frame with model estimates and actual data as columns.

models
[list] A list of models corresponding to estimated treatment effect columns.

outcome_col
[str][optional (default=y)] The column name for the actual outcome.

treatment_col
[str][optional (default=w)] The column name for the treatment indicator (0 or 1).

treatment_effect_col
[str][optional (default=tau)] The column name for the true treatment effect.

normalize
[bool][not implemented (default=False)] For consitency with gain and qini.

random_seed
[int, optional (default=None)] Random seed for numpy.random.rand().

Returns

effects
[pandas.DataFrame] Average causal effects of model estimates in cumulative population.

causeinfer.evaluation.**get_cum_gain**(*df*, *models=None*, *outcome_col='y'*, *treatment_col='w'*, *treatment_effect_col='tau'*, *normalize=False*, *random_seed=None*)

Gets cumulative gains of model estimates in population.

Parameters

df
[pandas.DataFrame] A data frame with model estimates and actual data as columns.

models
[list] A list of models corresponding to estimated treatment effect columns.

outcome_col
[str][optional (default=y)] The column name for the actual outcome.

treatment_col
[str][optional (default=w)] The column name for the treatment indicator (0 or 1).

treatment_effect_col
[str][optional (default=tau)] The column name for the true treatment effect.

normalize
[bool][optional (default=False)] Whether to normalize the y-axis to 1 or not.

random_seed
[int, optional (default=None)] Random seed for numpy.random.rand().

Returns

gains
[pandas.DataFrame] Cumulative gains of model estimates in population.

causeinfer.evaluation.**get_qini**(*df*, *models=None*, *outcome_col='y'*, *treatment_col='w'*, *treatment_effect_col='tau'*, *normalize=False*, *random_seed=None*)

Gets Qini of model estimates in population.

Parameters

df
[pandas.DataFrame] A data frame with model estimates and actual data as columns.

models
[list] A list of models corresponding to estimated treatment effect columns.

outcome_col
[str][optional (default=y)] The column name for the actual outcome.

treatment_col
[str][optional (default=w)] The column name for the treatment indicator (0 or 1).

treatment_effect_col
[str][optional (default=tau)] The column name for the true treatment effect.

normalize
[bool][optional (default=False)] Whether to normalize the y-axis to 1 or not.

random_seed
[int, optional (default=None)] Random seed for numpy.random.rand().

Returns

qinis
[pandas.DataFrame] Qini of model estimates in population.

causeinfer.evaluation.**auuc_score**(*df*, *models=None*, *outcome_col='y'*, *treatment_col='w'*, *treatment_effect_col='tau'*, *normalize=False*, *random_seed=None*)

Calculates the AUUC score (Gini): the Area Under the Uplift Curve.

Parameters

df
[pandas.DataFrame] A data frame with model estimates and actual data as columns.

models
[list] A list of models corresponding to estimated treatment effect columns.

outcome_col
[str][optional (default=y)] The column name for the actual outcome.

treatment_col
[str][optional (default=w)] The column name for the treatment indicator (0 or 1).

treatment_effect_col
[str][optional (default=tau)] The column name for the true treatment effect.

normalize
[bool][optional (default=False)] Whether to normalize the y-axis to 1 or not.

random_seed
[int, for inheritance (default=None)] Random seed for numpy.random.rand().

Returns

AUUC score
[float]

causeinfer.evaluation.**qini_score**(*df*, *models=None*, *outcome_col='y'*, *treatment_col='w'*, *treatment_effect_col='tau'*, *normalize=False*, *random_seed=None*)

Calculates the Qini score: the area between the Qini curve of a model and random assignment.

Parameters

df
[pandas.DataFrame)] A data frame with model estimates and actual data as columns

models
[list] A list of models corresponding to estimated treatment effect columns.

outcome_col
[str][optional (default=y)] The column name for the actual outcome.

treatment_col
[str][optional (default=w)] The column name for the treatment indicator (0 or 1).

treatment_effect_col
[str][optional (default=tau)] The column name for the true treatment effect.

normalize
[bool][optional (default=False)] Whether to normalize the y-axis to 1 or not.

random_seed
[int, for inheritance (default=None)] Random seed for numpy.random.rand().

Returns

Qini score
[float]

causeinfer.evaluation.**signal_to_noise**(*y*, *w*)

Computes the signal to noise ratio of a dataset to derive the potential for causal inference efficacy.

Parameters

y
[numpy.ndarray][(num_units,)][int, float] Vector of unit responses.

w
[numpy.ndarray][(num_units,)][int, float] Vector of original treatment allocations across units.

Returns

sn_ratio
[float]

Notes

- The signal to noise ratio is the difference in treatment and control response to the control response.
- Values close to 0 imply that CI would have little benefit over predictive modeling.

2.2 Plots

causeinfer plots provide graphical representations of model performance.

Functions

- *causeinfer.evaluation.plot_eval()*
- *causeinfer.evaluation.plot_cum_effect()*
- *causeinfer.evaluation.plot_cum_gain()*
- *causeinfer.evaluation.plot_qini()*
- *causeinfer.evaluation.plot_batch_metrics()*
- *causeinfer.evaluation.plot_batch_responses()*

causeinfer.evaluation.**plot_eval**(*df*, *kind=None*, *n=100*, *percent_of_pop=False*, *normalize=False*, *figsize=(15, 5)*, *fontsize=20*, *axis=None*, *legend_metrics=None*, **args*, ***kwargs*)

Plots one of the effect/gain/qini charts of model estimates.

Parameters

df
[pandas.DataFrame] A data frame with model estimates and unit outcomes as columns.

kind
[str][optional (default='gain')] The kind of plot to draw: 'effect,' 'gain,' and 'qini' are supported.

n
[int, optional (default=100)] The number of samples to be used for plotting.

percent_of_pop
[bool][optional (default=False)] Whether the X-axis is displayed as a percent of the whole population.

normalize
[bool][for inheritance (default=False)] Passes this argument to interior functions directly.

figsize
[tuple][optional] Allows for quick changes of figures sizes.

fontsize
[int or float][optional (default=20)] The font size of the plots, with all labels scaled accordingly.

axis
[str][optional (default=None)] Adds an axis to the plot so they can be combined.

legend_metrics
[bool][optional (default=True)] Calculate AUUC or Qini metrics to add to the plot legend for gain and qini respectively.

causeinfer.evaluation.**plot_cum_effect**(*df*, *n=100*, *models=None*, *percent_of_pop=False*, *outcome_col='y'*, *treatment_col='w'*, *treatment_effect_col='tau'*, *random_seed=None*, *figsize=None*, *fontsize=20*, *axis=None*, *legend_metrics=None*)

Plots the causal effect chart of model estimates in cumulative population.

Parameters

df
[pandas.DataFrame] A data frame with model estimates and actual data as columns.

kind
[effect] The kind of plot to draw

n
[int, optional (default=100)] The number of samples to be used for plotting.

models
[list] A list of models corresponding to estimated treatment effect columns.

percent_of_pop
[bool][optional (default=False)] Whether the X-axis is displayed as a percent of the whole population.

outcome_col
[str][optional (default=y)] The column name for the actual outcome.

treatment_col
[str][optional (default=w)] The column name for the treatment indicator (0 or 1).

treatment_effect_col
[str][optional (default=tau)] The column name for the true treatment effect.

random_seed
[int, optional (default=None)] Random seed for numpy.random.rand().

figsize
[tuple][optional] Allows for quick changes of figures sizes.

fontsize
[int or float][optional (default=20)] The font size of the plots, with all labels scaled accordingly.

axis
[str][optional (default=None)] Adds an axis to the plot so they can be combined.

legend_metrics
[bool][optional (default=False)] Not supported for plot_cum_effect - the user will be notified.

Returns

A plot of the cumulative effects of all models in df.

causeinfer.evaluation.**plot_cum_gain**(*df*, *n=100*, *models=None*, *percent_of_pop=False*, *outcome_col='y'*, *treatment_col='w'*, *treatment_effect_col='tau'*, *normalize=False*, *random_seed=None*, *figsize=None*, *fontsize=20*, *axis=None*, *legend_metrics=True*)

Plots the cumulative gain chart (or uplift curve) of model estimates.

Parameters

df
[pandas.DataFrame] A data frame with model estimates and actual data as columns.

kind
[gain] The kind of plot to draw

n
[int, optional (default=100)] The number of samples to be used for plotting.

models
[list] A list of models corresponding to estimated treatment effect columns.

percent_of_pop
[bool][optional (default=False)] Whether the X-axis is displayed as a percent of the whole population.

outcome_col
[str][optional (default=y)] The column name for the actual outcome.

treatment_col
[str][optional (default=w)] The column name for the treatment indicator (0 or 1).

treatment_effect_col
[str][optional (default=tau)] The column name for the true treatment effect.

normalize
[bool][optional (default=False)] Whether to normalize the y-axis to 1 or not.

random_seed
[int, optional (default=None)] Random seed for numpy.random.rand().

figsize
[tuple][optional] Allows for quick changes of figures sizes.

fontsize
[int or float][optional (default=20)] The font size of the plots, with all labels scaled accordingly.

axis
[str][optional (default=None)] Adds an axis to the plot so they can be combined.

legend_metrics
[bool][optional (default=True)] Calculates AUUC metrics to add to the plot legend.

Returns

A plot of the cumulative gains of all models in df.

causeinfer.evaluation.**plot_qini**(*df*, *n=100*, *models=None*, *percent_of_pop=False*, *outcome_col='y'*, *treatment_col='w'*, *treatment_effect_col='tau'*, *normalize=False*, *random_seed=None*, *figsize=None*, *fontsize=20*, *axis=None*, *legend_metrics=True*)

Plots the Qini chart (or uplift curve) of model estimates.

Parameters

df
[pandas.DataFrame] A data frame with model estimates and actual data as columns.

kind
[qini] The kind of plot to draw

n
[int, optional (default=100)] The number of samples to be used for plotting.

models
[list] A list of models corresponding to estimated treatment effect columns.

percent_of_pop
[bool][optional (default=False)] Whether the X-axis is displayed as a percent of the whole population.

outcome_col
[str][optional (default=y)] The column name for the actual outcome.

treatment_col
[str][optional (default=w)] The column name for the treatment indicator (0 or 1).

treatment_effect_col
[str][optional (default=tau)] The column name for the true treatment effect.

normalize
[bool][optional (default=False)] Whether to normalize the y-axis to 1 or not.

random_seed
[int, optional (default=None)] Random seed for numpy.random.rand().

figsize
[tuple][optional] Allows for quick changes of figures sizes.

fontsize
[int or float][optional (default=20)] The font size of the plots, with all labels scaled accordingly.

axis
[str][optional (default=None)] Adds an axis to the plot so they can be combined.

legend_metrics
[bool][optional (default=True)] Calculates Qini metrics to add to the plot legend.

Returns

A plot of the qini curves of all models in df.

causeinfer.evaluation.**plot_batch_metrics**(*df*, *kind=None*, *n=10*, *models=None*, *outcome_col='y'*, *treatment_col='w'*, *normalize=False*, *figsize=(15, 5)*, *fontsize=20*, *axis=None*, **args*, ***kwargs*)

Plots the batch chart: the cumulative batch metrics predicted by a model given ranked treatment effects.

Parameters

df
[pandas.DataFrame] A data frame with model estimates and unit outcomes as columns.

kind
[str][optional (default='gain')] The kind of plot to draw: 'effect,' 'gain,' 'qini,' and 'response' are supported.

n
[int, optional (default=10, deciles; 20, quintiles also standard)] The number of batches to split the units into.

models
[list] A list of models corresponding to estimated treatment effect columns.

outcome_col
[str][optional (default=y)] The column name for the actual outcome.

treatment_col
[str][optional (default=w)] The column name for the treatment indicator (0 or 1).

figsize
[tuple][optional] Allows for quick changes of figures sizes.

fontsize
[int or float][optional (default=20)] The font size of the plots, with all labels scaled accordingly.

axis
[str][optional (default=None)] Adds an axis to the plot so they can be combined.

Returns

A plot of batch metrics of all models in df.

causeinfer.evaluation.**plot_batch_responses**(*df*, *n=10*, *models=None*, *outcome_col='y'*, *treatment_col='w'*, *normalize=False*, *figsize=(15, 5)*, *fontsize=20*, *axis=None*)

Plots the batch response chart: the cumulative batch responses predicted by a model given ranked treatment effects.

Parameters

df
[pandas.DataFrame] A data frame with model estimates and unit outcomes as columns.

kind
[response] The kind of plot to draw

n
[int, optional (default=10, deciles; 20, quintiles also standard)] The number of batches to split the units into.

models
[list] A list of models corresponding to estimated treatment effect columns.

outcome_col
[str][optional (default=y)] The column name for the actual outcome.

treatment_col
[str][optional (default=w)] The column name for the treatment indicator (0 or 1).

figsize
[tuple][optional] Allows for quick changes of figures sizes.

fontsize
[int or float][optional (default=20)] The font size of the plots, with all labels scaled accordingly.

axis
[str][optional (default=None)] Adds an axis to the plot so they can be combined.

Returns

A plot of batch responses of all models in df.

2.3 Iteration

Iterations methods allow a researcher or practitioner to derive average model accuracy.

Functions

- *causeinfer.evaluation.iterate_model()*
- *causeinfer.evaluation.eval_table()*

causeinfer.evaluation.**iterate_model**(*model*, *X_train*, *y_train*, *w_train*, *X_test*, *y_test*, *w_test*, *tau_test=None*, *n=10*, *pred_type='predict'*, *eval_type=None*, *normalize_eval=False*, *verbose=True*)

Trains and makes predictions with a model multiple times to derive average predictions and their variance.

Parameters

model
[object] A model over which iterations will be done.

X_train
[numpy.ndarray][(num_train_units, num_features)][int, float] Matrix of covariates.

y_train
[numpy.ndarray][(num_train_units,)][int, float] Vector of unit responses.

w_train
[numpy.ndarray][(num_train_units,)][int, float] Vector of original treatment allocations across units.

X_test
[numpy.ndarray][(num_test_units, num_features)][int, float] A matrix of covariates.

y_test
[numpy.ndarray][(num_test_units,)][int, float] A vector of unit responses.

w_test
[numpy.ndarray][(num_test_units,)][int, float] A vector of original treatment allocations across units.

tau_test
[numpy.ndarray][(num_test_units,)][int, float] A vector of the actual treatment effects given simulated data.

n
[int (default=10)] The number of train and prediction iterations to run.

pred_type
[str (default=pred)] predict or predict_proba: the type of prediction the iterations will make.

eval_type
[str (default=None)] qini or auuc: the type of evaluation to be done on the predictions.

Note: if None, model predictions will be averaged without their variance being calculated.

normalize_eval
[bool][optional (default=False)] Whether to normalize the evaluation metric.

verbose
[bool (default=True)] Whether to show a tqdm progress bar for the query.

Returns

avg_preds_probas
[numpy.ndarray (num_units, 2)][float] Averaged per unit predictions.

all_preds_probas
[dict] A dictionary of all predictions produced during iterations.

avg_eval
[float] The average of the iterated model evaluations.

eval_variance
[float] The variance of all prediction evaluations.

eval_variance
[float] The variance of all prediction evaluations.

all_evals
[dict] A dictionary of all evaluations produced during iterations.

causeinfer.evaluation.**eval_table**(*eval_dict*, *variances=False*, *annotate_vars=False*)

Displays the evaluation of models given a dictionary of their evaluations over datasets.

Parameters

eval_dict
[dict] A dictionary of model evaluations over datasets.

variances
[bool (default=False)] Whether to annotate the evaluations with their variances.

annotate_vars
[bool (default=False)] Whether to annotate the evaluation variances with stars given their sds.

Returns

eval_table
[pandas.DataFrame][(num_datasets, num_models)] A dataframe of dataset to model evaluation comparisons.

CHAPTER

THREE

DATA

Data in causeinfer provides examples for business, medical, and socio-economic fields as benchmarks for CI techniques.

3.1 Hillstrom Email Marketing

An email marketing dataset from Kevin Hillstrom's MineThatData blog.

See an example using this data at causeinfer/examples/business_hillstrom.

Description found at:
https://blog.minethatdata.com/2008/03/minethatdata-e-mail-analytics-and-data.html

Based on

Kuchumov, A. pyuplift: Lightweight uplift modeling framework for Python. (2019). URL: https://github.com/duketemon/pyuplift. License: https://github.com/duketemon/pyuplift/blob/master/LICENSE.

K. Hillstrom. "The MineThatData E-Mail Analytics And Data Mining Challenge". 2008. URL: https://blog.minethatdata.com/2008/03/minethatdata-e-mail-analytics-and-data.html.

Contents

download_hillstrom, _format_data, load_hillstrom

causeinfer.data.hillstrom.**download_hillstrom**(*data_path=None*, *url='http://www.minethatdata.com/Kevin_Hillstrom_MineThatData_E-MailAnalytics_DataMiningChallenge_2008.03.20.csv'*)

Downloads the dataset from Kevin Hillstrom's blog.

Parameters

data_path
[str][optional (default=None)] A user specified path for where the data should go.

url
[str] The url from which the data is to be downloaded.

Returns

The data 'hillstrom.csv' in a 'datasets' folder, unless otherwise specified.

causeinfer.data.hillstrom.**_format_data**(*df*, *format_covariates=True*, *normalize=True*)

Formats the data upon loading for consistent data preparation.

Parameters

df
[pd.DataFrame] The original unformatted version of the data.

format_covariates
[bool][optional (default=True), controlled in load_hillstrom]

- True: creates dummy columns and encodes the data.
- False: only steps for data readability will be taken.

normalize
[bool][optional (default=True), controlled in load_hillstrom] Normalize dataset columns to prepare them for ML methods.

Returns

df
[pd.DataFrame] A formated version of the data.

causeinfer.data.hillstrom.**load_hillstrom**(*file_path=None*, *format_covariates=True*, *download_if_missing=True*, *normalize=True*)

Loads the Hillstrom dataset with formatting if desired.

Parameters

file_path
[str][optional (default=None)] Specify another path for the dataset.

By default the dataset should be stored in the 'datasets' folder in the cwd.

format_covariates
[bool][optional (default=True)] Indicates whether raw data should be loaded without covariate manipulation.

download_if_missing
[bool][optional (default=True)] Download the dataset if it is not downloaded before using 'download_hillstrom'.

normalize
[bool][optional (default=True)] Normalize dataset columns to prepare them for ML methods.

Returns

data
[dict object with the following attributes:]

data.description
[str] A description of the Hillstrom email marketing dataset.

data.dataset_full
[numpy.ndarray][(64000, 12) or formatted (64000, 22)] The full dataset with features, treatment, and target variables.

data.dataset_full_names
[list, size 12 or formatted 22] List of dataset variables names.

data.features
[numpy.ndarray][(64000, 8) or formatted (64000, 18)] Each row corresponding to the 8 feature values in order.

data.feature_names
[list, size 8 or formatted 18] List of feature names.

data.treatment
[numpy.ndarray][(64000,)] Each value corresponds to the treatment.

data.response_spend
[numpy.ndarray][(64000,)] Each value corresponds to how much customers spent during the two-week outcome period.

data.response_visit
[numpy.ndarray][(64000,)] Each value corresponds to whether people visited the site during the two-week outcome period.

data.response_conversion
[numpy.ndarray][(64000,)] Each value corresponds to whether they purchased at the site (i.e. converted) during the two-week outcome period.

3.2 Mayo Clinic PBC

A dataset on medical trials to combat primary biliary cholangitis (PBC, formerly cirrhosis) of the liver from the Mayo Clinic.

See an example using this data at causeinfer/examples/medical_mayo_pbc.

Description found at:
https://www.mayo.edu/research/documents/pbchtml/DOC-10027635

Based on
Mayo Clinic. "Primary Biliary Cirrhosis". 1991. URL: https://www.mayo.edu/research/documents/pbchtml/DOC-10027635.

Contents
download_mayo_pbc, _format_data, load_mayo_pbc

causeinfer.data.mayo_pbc.**download_mayo_pbc**(*data_path=None*, *url='http://www.mayo.edu/research/documents/pbcdat/DOC-10026921'*)

Downloads the dataset from the Mayo Clinic's research documents.

Parameters

data_path
[str][optional (default=None)] A user specified path for where the data should go.

url
[str] The url from which the data is to be downloaded.

Returns

The text file 'mayo_pbc' in a 'datasets' folder, unless otherwise specified.

causeinfer.data.mayo_pbc.**_format_data**(*dataset_path*, *format_covariates=True*, *normalize=True*)

Formats the data upon loading for consistent data preparation.

Parameters

dataset_path
[str] The original file is a text file with inconsistent spacing, and periods for NaNs.

Furthermore, process only loads those units that took part in the randomized trial, as there are 106 cases that were monitored, but not in the trial.

format_covariates
[bool][optional (default=True)]

- True: creates dummy columns and encodes the data.

- False: only steps for data readability will be taken.

normalize
[bool][optional (default=True)] Normalization step controlled in load_mayo_pbc.

Returns

df
[pd.DataFrame] A formated version of the data.

causeinfer.data.mayo_pbc.**load_mayo_pbc**(*file_path=None*, *format_covariates=True*, *download_if_missing=True*, *normalize=True*)

Loads the Mayo PBC dataset with formatting if desired.

Parameters

file_path
[str][optional (default=None)] Specify another path for the dataset.

By default the dataset should be stored in the 'datasets' folder in the cwd.

format_covariates
[bool][optional (default=True)] Indicates whether raw data should be loaded without covariate manipulation.

download_if_missing
[bool][optional (default=True)] Download the dataset if it is not downloaded before using 'download_mayo_pbc'.

normalize
[bool][optional (default=True)] Normalize the dataset to prepare it for ML methods.

Returns

data
[dict object with the following attributes:]

data.description
[str] A description of the Mayo Clinic PBC dataset.

data.dataset_full
[numpy.ndarray][312, 19) or formatted (312, 24)] The full dataset with features, treatment, and target variables.

data.dataset_full_names
[list, size 19 or formatted 24] List of dataset variables names.

data.features
[numpy.ndarray][(312, 17) or formatted (312, 22)] Each row corresponding to the 17 feature values in order.

data.feature_names
[list, size 17 or formatted 22] List of feature names.

data.treatment
[numpy.ndarray][(312,)] Each value corresponds to the treatment (1 = treat, 0 = control).

data.response
[numpy.ndarray][(312,)] Each value corresponds to one of the outcomes (0 = alive, 1 = liver transplant, 2 = dead).

3.3 CMF Microfinance

A dataset on microfinance from The Centre for Micro Finance (CMF) at the Institute for Financial Management Research (Chennai, India).

See an example using this data at causeinfer/examples/socioeconomic_cmf_micro.

Description found at:
https://www.aeaweb.org/articles?id=10.1257/app.20130533 (see paper)

Based on
A. Banerjee et al. "The Miracle of Microfinance? Evidence from a Randomized Evaluation". In: American Economic Journal: Applied Economics 7 (1 2015), pp. 22–53. URL: https://www.aeaweb.org/articles?id=10.1257/app.20130533.

Contents
download_cmf_micro (deprecated), _format_data, load_cmf_micro

`causeinfer.data.cmf_micro._format_data`(*dataset_path*, *format_covariates=True*, *normalize=True*)

Formats the data upon loading for consistent data preparation.

Source: https://github.com/thmstang/apa19-microfinance/blob/master/helpers.r (R-version)

Parameters

dataset_path
[str] The original file is a folder that has various .dta sets.

format_covariates
[bool][optional (default=True)]

- True: creates dummy columns and encodes the data.
- False: only steps for data readability will be taken.

normalize
[bool][optional (default=True)] Normalization step controlled in load_cmf_micro.

Returns

df
[pd.DataFrame] A formated version of the data.

`causeinfer.data.cmf_micro.load_cmf_micro`(*file_path=None*, *format_covariates=True*, *normalize=True*)

Loads the CMF micro dataset with formatting if desired.

Parameters

file_path
[str][optional (default=None)] Specify another path for the dataset.

By default the dataset should be stored in the 'datasets' folder in the cwd.

load_raw_data
[bool][optional (default=True)] Indicates whether raw data should be loaded without covariate manipulation.

download_if_missing
[bool][optional (default=True) (Deprecated)] Download the dataset if it is not downloaded before using 'download_cmf_micro'.

normalize
[bool][optional (default=True)] Normalize the dataset to prepare it for ML methods.

Returns

data
[dict object with the following attributes:]

data.description
[str] A description of the CMF microfinance data.

data.dataset_full
[numpy.ndarray][(5328, 183) or formatted (5328, 60)] The full dataset with features, treatment, and target variables.

data.dataset_full_names
[list, size 61] List of dataset variables names.

data.features
[numpy.ndarray][(5328, 186) or formatted (5328, 57)] Each row corresponding to the 58 feature values in order (note that other target can be a feature).

data.feature_names
[list, size 58] List of feature names.

data.treatment
[numpy.ndarray][(5328,)] Each value corresponds to the treatment (1 = treat, 0 = control).

data.response_biz_index
[numpy.ndarray][(5328,)] Each value corresponds to the business index of each of the participants.

data.response_women_emp
[numpy.ndarray][(5328,)] Each value corresponds to the women's empowerment index of each of the participants.

3.4 Download Utilities

Utility functions for downloading data.

Based on
Kuchumov, A. pyuplift: Lightweight uplift modeling framework for Python. (2019). URL: https://github.com/duketemon/pyuplift. License: https://github.com/duketemon/pyuplift/blob/master/LICENSE.

Contents
download_file, get_download_paths

`causeinfer.data.download_utils.`**`download_file`**`(`*url: str*, *output_path: str*, *zip_file=False*`)`
Downloads a file from a url to a specified path.

Parameters

url
[str] the URL from which the file can be downloaded from.

output_path
[str] a user specified path, which defaults to a 'files' folder in the cwd.

`causeinfer.data.download_utils.`**`get_download_paths`**`(`*file_path*, *file_directory='files'*, *file_name='file'*`)`
Derives paths for a file folder and a file.

Parameters

path
[str] A user specified path that the data should go to

file_directory
[str (default=files)] A user specified directory.

file_name
[str (default=file)] The name to call the file.

CHAPTER

FOUR

UTILS

The `utils` module provides needed functions for causal inference model testing and deployment.

Functions

- *causeinfer.utils.train_test_split()*
- *causeinfer.utils.plot_unit_distributions()*
- *causeinfer.utils.over_sample()*
- causeinfer.utils.mutli_cross_tab()

causeinfer.utils.**train_test_split**(*X*, *y*, *w*, *percent_train=0.7*, *random_state=None*, *maintain_proportions=False*)

Split unit X covariates and (y,w) outcome tuples into training and testing sets.

Parameters

X
[numpy.ndarray][(n_samples, n_features)] Matrix of unit covariate features.

y
[numpy.ndarray][(n_samples,)] Array of unit responses.

w
[numpy.ndarray][(n_samples,)] Array of unit treatments.

percent_train
[float] The percent of the covariates and outcomes to delegate to model training.

random_state
[int (default=None)] A seed for the random number generator for consistency.

maintain_proportions
[bool][optional (default=False)] Whether to maintain the treatment group proportions within the split samples.

Returns

X_train, X_test, y_train, y_test, w_train, w_test
[numpy.ndarray] Arrays of split covariates and outcomes.

causeinfer.utils.**plot_unit_distributions**(*df*, *variable*, *treatment=None*, *bins=None*, *axis=None*)

Plots seaborn countplots of unit covariate and outcome distributions.

Parameters

df_plot
[pandas df, [n_samples, n_features]] The data from which the plot is made.

variable
[str] A unit covariate or outcome for which the plot is desired.

treatment
[str][optional (default=None)] The treatment variable for comparing across segments.

bins
[int (default=None)] Bins the column values such that larger distributions can be plotted.

axis
[str][optional (default=None)] Adds an axis to the plot so they can be combined.

Returns

ax
[matplotlib.axes] Displays a seaborn plot of unit distributions across the given covariate or outcome value.

causeinfer.utils.**over_sample**(*X_1*, *y_1*, *w_1*, *sample_2_size*, *shuffle=True*, *random_state=None*)

Over-samples to provide equality between a given sample and another it is smaller than.

Parameters

X_1
[numpy.ndarray][(num_sample1_units, num_sample1_features)] Dataframe of sample covariates.

y_1
[numpy.ndarray][(num_sample1_units,)] Vector of sample unit responses.

w_1
[numpy.ndarray][(num_sample1_units,)] Designates the original treatment allocation across sample units.

sample_2_size
[int] The size of the other sample to match.

shuffle
[bool][optional (default=True)] Whether to shuffle the new sample after it's created.

random_state
[int (default=None)] A seed for the random number generator to allow for consistency.

Returns

The provided covariates and outcomes, having been over-sampled to match another.

- X_os : numpy.ndarray : (num_sample2_units, num_sample2_features).
- y_os : numpy.ndarray : (num_sample2_units,).
- w_os : numpy.ndarray : (num_sample2_units,).

CHAPTER

FIVE

CONTRIBUTING TO CAUSEINFER

Thank you for your consideration in contributing to this project!

Please take a moment to review this document in order to make the contribution process easy and effective for everyone involved.

Following these guidelines helps to communicate that you respect the time of the developers managing and developing this open source project. In return, and in accordance with this project's code of conduct, other contributors will reciprocate that respect in addressing your issue or assessing patches and features.

5.1 Using the issue tracker

The issue tracker for causeinfer is the preferred channel for *bug reports*, *features requests* and *submitting pull requests*.

5.2 Bug reports

A bug is a *demonstrable problem* that is caused by the code in the repository. Good bug reports are extremely helpful - thank you!

Guidelines for bug reports:

1. **Use the GitHub issue search** to check if the issue has already been reported.
2. **Check if the issue has been fixed** by trying to reproduce it using the latest `main` or development branch in the repository.
3. **Isolate the problem** to make sure that the code in the repository is *definitely* responsible for the issue.

Great Bug Reports tend to have:

- A quick summary
- Steps to reproduce
- What you expected would happen
- What actually happens
- Notes (why this might be happening, things tried that didn't work, etc)

Again, thank you for your time in reporting issues!

5.3 Feature requests

Feature requests are more than welcome! Please take a moment to find out whether your idea fits with the scope and aims of the project. When making a suggestion, provide as much detail and context as possible, and further make clear the degree to which you would like to contribute in its development.

5.4 Pull requests

Good pull requests - patches, improvements and new features - are a fantastic help. They should remain focused in scope and avoid containing unrelated commits. Note that all contributions to this project will be made under the specified license and should follow the coding indentation and style standards (contact us if unsure).

Please ask first before embarking on any significant pull request (implementing features, refactoring code, etc), otherwise you risk spending a lot of time working on something that the developers might not want to merge into the project. With that being said, major additions are very appreciated!

When making a contribution, adhering to the GitHub flow process is the best way to get your work merged:

1. Fork the repo, clone your fork, and configure the remotes:

```
# Clone your fork of the repo into the current directory
git clone https://github.com/<your-username>/<repo-name>
# Navigate to the newly cloned directory
cd <repo-name>
# Assign the original repo to a remote called "upstream"
git remote add upstream https://github.com/<upsteam-owner>/<repo-name>
```

2. If you cloned a while ago, get the latest changes from upstream:

```
git checkout <dev-branch>
git pull upstream <dev-branch>
```

3. Create a new topic branch (off the main project development branch) to contain your feature, change, or fix:

```
git checkout -b <topic-branch-name>
```

4. Commit your changes in logical chunks, and please try to adhere to Conventional Commits. Use Git's interactive rebase feature to tidy up your commits before making them public.

5. Locally merge (or rebase) the upstream development branch into your topic branch:

```
git pull --rebase upstream <dev-branch>
```

6. Push your topic branch up to your fork:

```
git push origin <topic-branch-name>
```

7. Open a Pull Request with a clear title and description.

Thank you in advance for your contributions!

5.5 License

5.6 Change log

CHAPTER

SIX

CHANGELOG

causeinfer tries to follow semantic versioning, a MAJOR.MINOR.PATCH version where increments are made of the:

- MAJOR version when we make incompatible API changes
- MINOR version when we add functionality in a backwards compatible manner
- PATCH version when we make backwards compatible bug fixes

CHAPTER

SEVEN

CAUSEINFER 1.0.1 (JUNE 3RD, 2022)

- Updates source code files with direct references to codes they're based on.

CHAPTER

EIGHT

CAUSEINFER 1.0.0 (DECEMBER 28TH, 2021)

- Release switches causeinfer over to semantic versioning and indicates that it is stable

CHAPTER

NINE

CAUSEINFER 0.1.2 (APRIL 4TH, 2021)

Changes include:

- An src structure has been adopted to improve organization and testing
- Users are now able to implement the following models:
  - Reflective Uplift (Shaar 2016)
  - Pessimistic Uplift (Shaar 2016)
- The contribution guidelines have been expanded
- Code quality checks via Codacy have been added
- Extensive code formatting has been done to improve quality and style
- Bug fixes and a more explicit use of exceptions

CHAPTER

TEN

CAUSEINFER 0.1.0 (FEB 25TH, 2021)

First stable release of causeinfer

- Users are able to implement baseline causal inference models including:
  - Two model
  - Interaction term (Lo 2002)
  - Binary transformation (Lai 2006)
  - Quaternary transformation (Kane 2014)
- Plotting functions allow for graphical analysis of models
- Functions useful for research such as model iterations, oversampling, and variance analysis are included
- The package is fully documented
- Virtual environment files are provided
- Extensive testing of all modules with GH Actions and Codecov has been performed
- A code of conduct and contribution guidelines are included

CHAPTER

ELEVEN

PROJECT INDICES

- genindex

PYTHON MODULE INDEX

C

INDEX

Symbols

A

B

C

D

E

F

R

S

T