Introduction to CausalML

Aug 2021

Presenter: Paul Lo[1], Totte Harinen[2]

Uber

CausalML

[1] Uber Technologies [2] Toyota Research Institute

Agenda

01 Why CausalML?

- Motivation and Use cases
- Highlights and Functionality overview

02 Get started with CausalML

03 Main modules

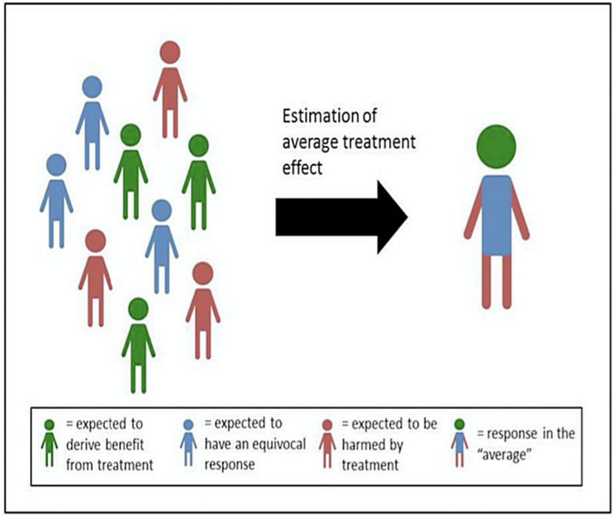
- Meta-learners and Uplift trees models
- Value optimization models
- Interpretation & Visualisation
- Data Generation and PSM

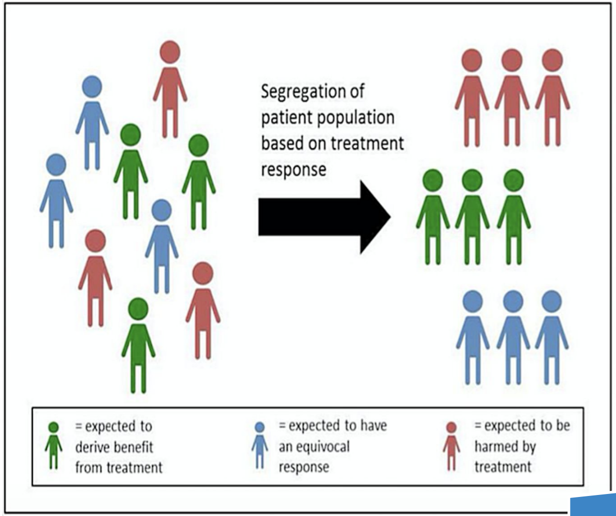
04 Summary

Installation: https://github.com/uber/causalml#installation
Notebook: https://github.com/uber/causalml/tree/master/examples
KDD website: https://causal-machine-learning.github.io/kdd2021-tutorial/

Why CausalML

Why CausalML | Motivation





From ATE to CATE

Enables personalized experience &
Optimize for incremental effects

Machine Learning + Causal Inference

Develop optimization applications
based on the unlocked insights

Graph credit: Xie, Yuxiang, Nanyu Chen, and Xiaolin Shi. "False discovery rate controlled heterogeneous treatment effect detection for online controlled experiments." Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018.

Why CausalML | Real-World Applications

Personalization

We can engage with users more effectively by learning heterogeneous treatment effect, for example, provide meaningful and proactive customer support

Causal Impact Analysis

In User Action and Customer Value Analysis project, we measure the impact of cross-sell conversions to existing users' long-term value.

Case Study #1: CeViChE

Budget Optimization

Ads Targeting: we can use uplift modeling to optimize return on ads spend by selecting persuadable user group. **Case Study #2: Bidder**

Promo Targeting: we use causal inference estimation to spend budget on users who is estimated to have better treatment effect

Why CausalML | One-stop shop of Machine Learning for Causal Inference

Dedicated support

It’s incubated for **long-term support** from Uber and community developers, we continue working actively with users to maintain and develop new functions and models.

Ease of Use

We follow **standardized machine learning library interface**. It’s super easy to onboard and develop your use case with few lines of code. Also, **extensibility for growth is our package design principle.**

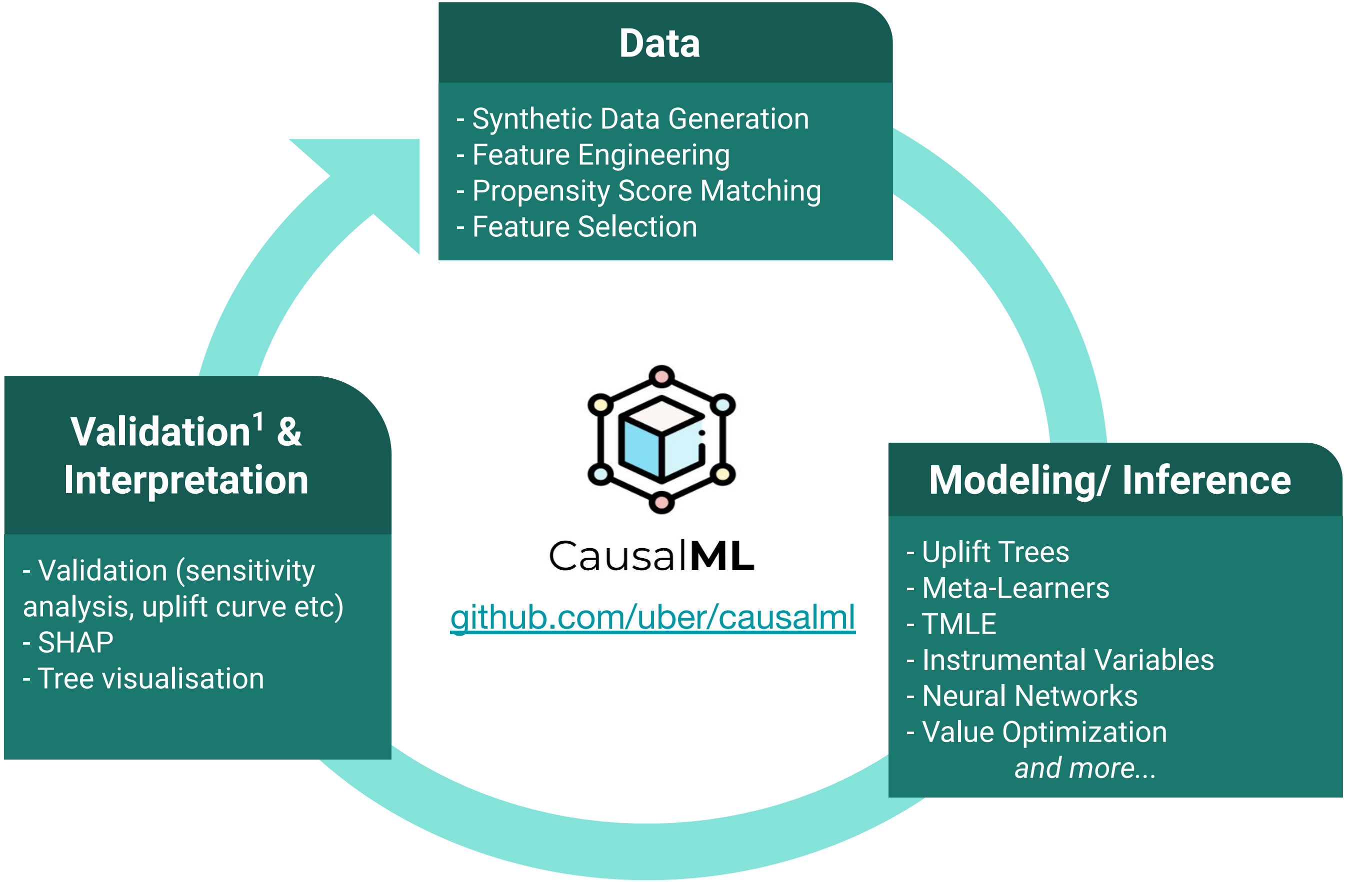
Industrial Use Cases

We aim to tackle **real-world large scale data and industrial applications** and continue to improve the efficiency of model implementation. We welcome collaborations on package development & use cases!

Rich Features

CausalML provides functions ranging from data generating, to PSM, to tree-based and meta-learners models. We **democratise uplift modeling methods** in academic paper and continue to add innovative in-house algorithms!

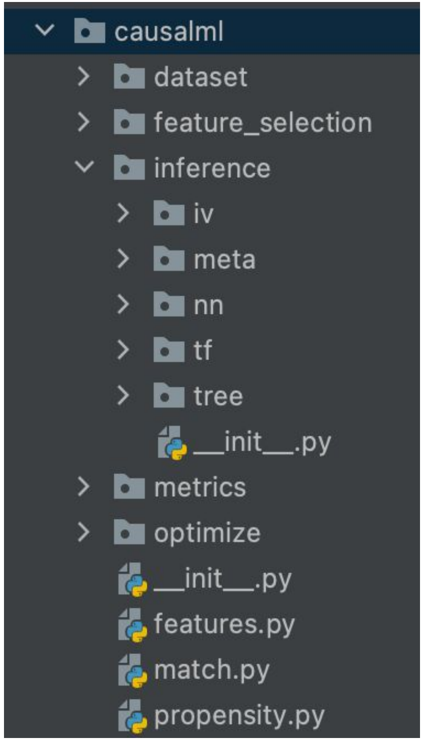
Why CausalML | Overview



[1] *Validation in Causal Inference: https://causalml.readthedocs.io/en/latest/validation.html*

Why CausalML | Overview

Package structure



Modules and Key Algorithms

Modules

- Data Generation Process
- Feature Engineering
- Propensity Score Methods
- Sensitivity Analysis
- Causal Inference models
- Visualization

Uplift Trees

- KL Divergence
- Chi-Square
- Euclidean Distance
- Contextual Treatment Selection

Meta Learners

- R-Learner
- X-Learner
- T-Learner
- S-Learner
- DR-Learner
- TMLE

Neural Networks

- DragonNet
- CEVAE

Instrumental Variables

- 2SLS
- Doubly Robust IV

Get Started with CausalML

Get started | Package installation (v0.11)

Local with Conda

Recommend to use conda

```
$ git clone https://github.com/uber/causalml.git
$ cd causalml/envs/
$ conda env create -f environment-py38.yml
$ conda activate causalml-py38
```

For tensorflow (required for DragonNet model)

```
$ git clone https://github.com/uber/causalml.git
$ cd causalml/envs/
$ conda env create -f environment-tf-py38.yml
$ conda activate causalml-tf-py38
(causalml-tf-py38) pip install -U numpy
```

Google Colab (or other cloud instance)

```
!pip install causalml
```

For tensorflow

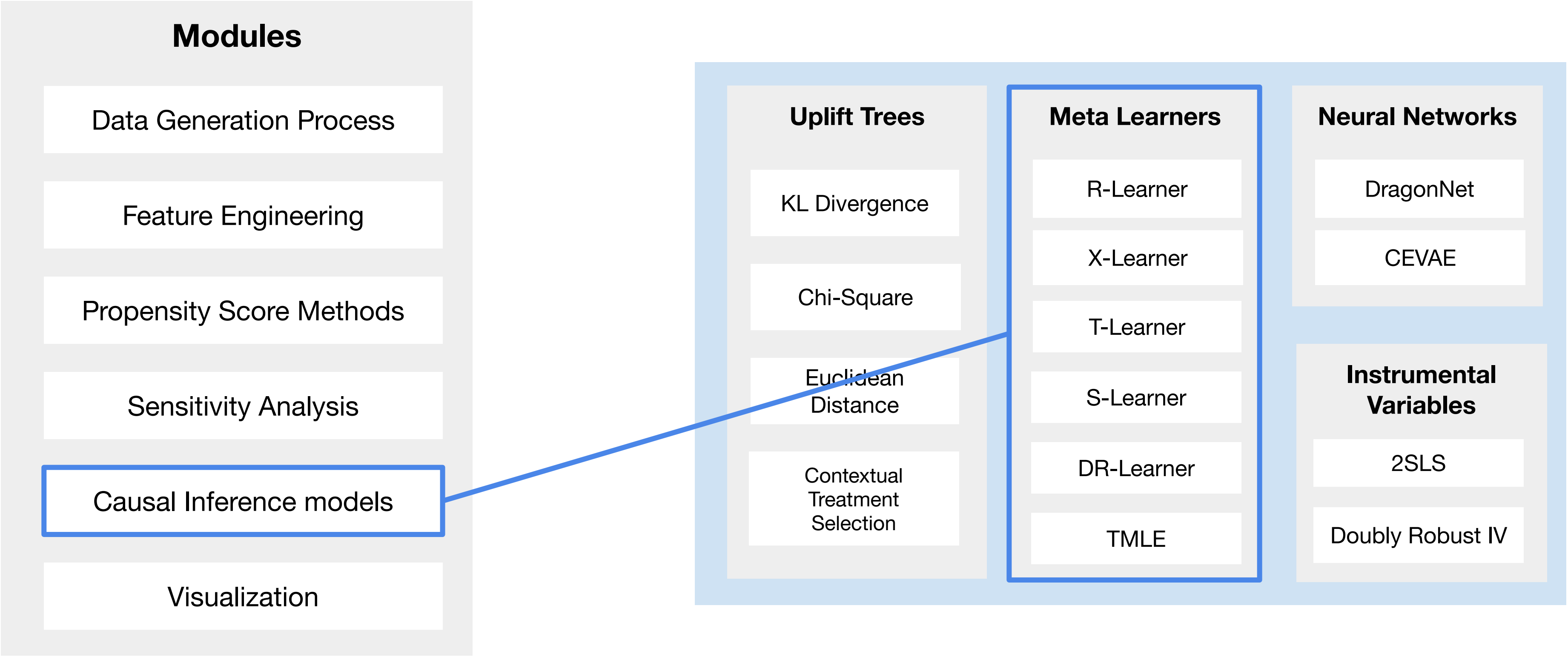
```
!pip install causalml[tf]
```

- *Support python 3.6, 3.7, 3.8, 3.9*
- *Package installation*
  https://github.com/uber/causalml#installation
- *Notebook example*
  https://github.com/uber/causalml/tree/master/examples

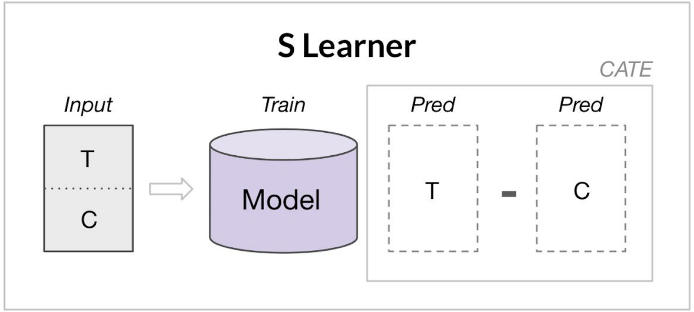
Main Modules

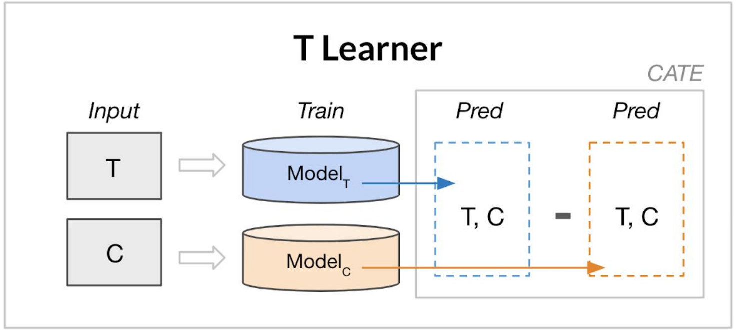
Meta-learners and Uplift Trees
Value Optimization
Visualization
Data Generation and PSM

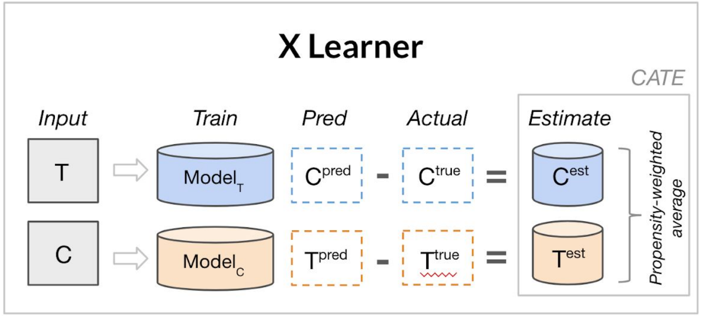
CausalML | Overview

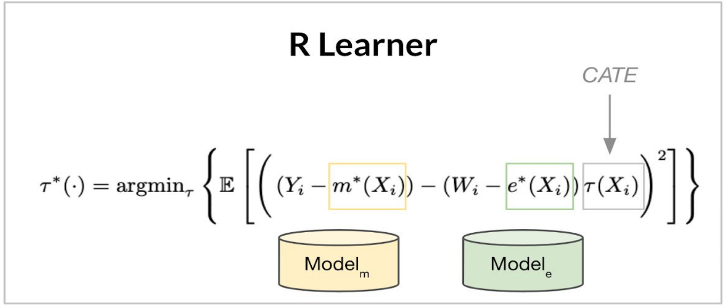


CausalML | Meta-learners: methodology



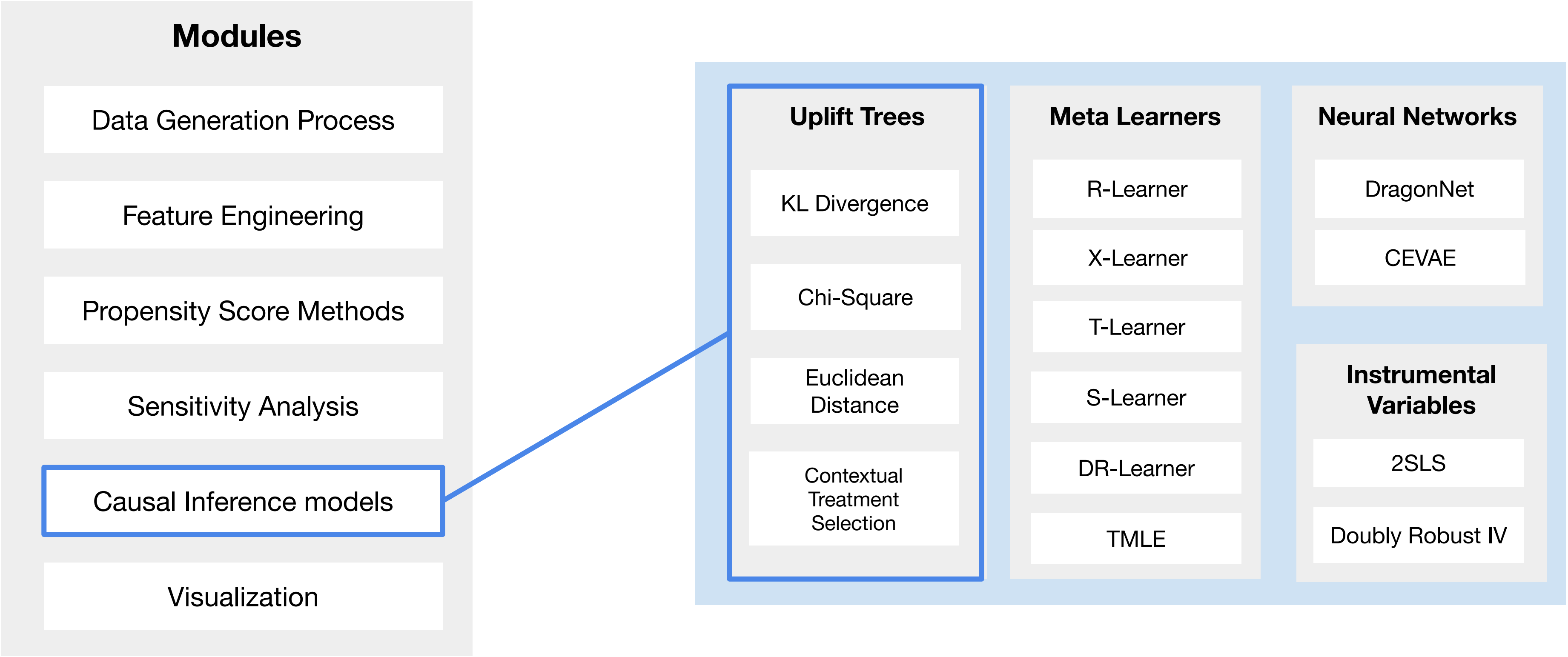






Künzel, Sören R., et al. "Metalearners for estimating heterogeneous treatment effects using machine learning." Proceedings of the national academy of sciences 116.10 (2019): 4156-4165.
Nie, Xinkun, and Stefan Wager. "Quasi-oracle estimation of heterogeneous treatment effects." arXiv preprint arXiv:1712.04912 (2017).

CausalML | Overview



CausalML | Uplift Trees: methodology

Classification Tree

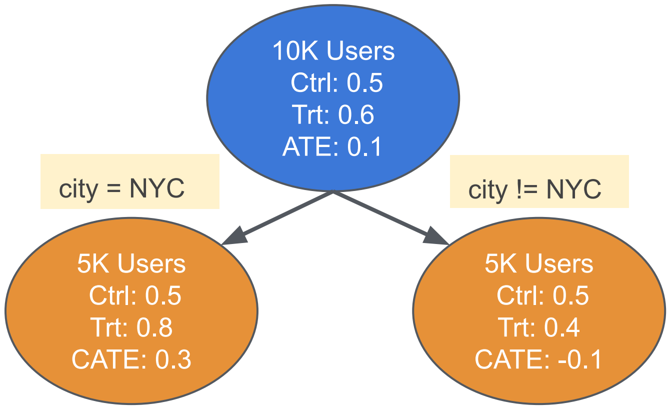
Goal: Predicting the conversion rate for the user

Loss function for tree split: The divergence with respect to the conversion rate. A split is great if the left note and right note has quite different conversion rate.

Uplift Tree

Goal: Predicting the treatment effect for the user

Loss function for tree split: The divergence with respect to the treatment effect. A split is great if the left note and the right note has quite different treatment effect.



Example loss: KL divergence

$$D_{KL}(P_t||P_c) = P_t(Y=1)log(\frac{P_t(Y=1)}{P_c(Y=1)}) + P_t(Y=0)log(\frac{P_t(Y=0)}{P_c(Y=0)})$$

Uplift Tree example:
[examples/uplift_trees_with_synthetic_data.ipynb]

CausalML | CATE estimation example

Conditional Average Treatment Effect (CATE)

```python
from causalml.inference.meta import BaseSRegressor
from xgboost import XGBRegressor

y, X, treatment, _, _, e = synthetic_data(mode=1, n=n-samples, p=n_features)

learner_s = BaseSRegressor(learner=XGBRegressor())
cate_s = learner_s.fit_predict(X=X, treatment=treatment, y=y)
```

Generate data

Initialize learner
Fit and predict

S/ T/ X/ R Learner

```python
from causalml.inference.meta import BaseSRegressor, BaseTRegressor, BaseXRegressor, BaseRRegressor

learner_s = LRSRegressor(learner=XGBRegressor())
learner_t = BaseTRegressor(learner=XGBRegressor())
learner_x = BaseXRegressor(learner=XGBRegressor())
learner_r = BaseRRegressor(learner=XGBRegressor())
cate_s = learner_s.fit_predict(X=X, treatment=treatment, y=y)
cate_t = learner_t.fit_predict(X=X, treatment=treatment, y=y)
cate_x = learner_x.fit_predict(X=X, treatment=treatment, y=y, p=e)
cate_r = learner_r.fit_predict(X=X, treatment=treatment, y=y, p=e)
```

Initialize learner

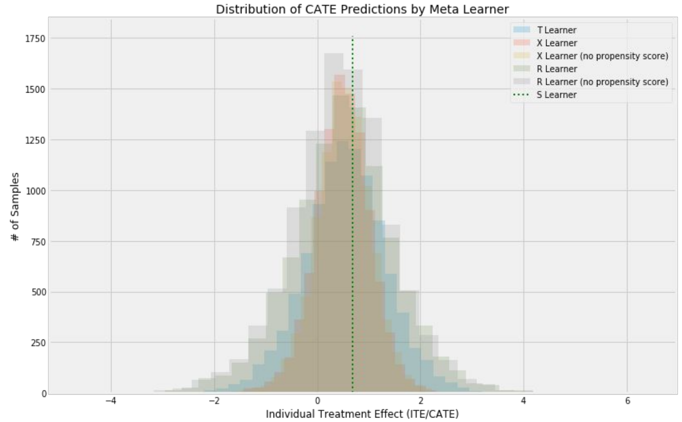
Fit and predict

Meta-Learners example:
examples/meta_learners_with_synthetic_data.ipynb

CausalML | CATE estimation example

Plot CATE predictions by learner

```python
plt.hist(cate_t, alpha=alpha, bins=bins, label='T Learner')
plt.hist(cate_x, alpha=alpha, bins=bins, label='X Learner')
plt.hist(cate_r, alpha=alpha, bins=bins, label='R Learner')
plt.vlines(cate_s[0], 0, plt.axes().get_ylim()[1], ... .)
plt.title('Distribution of CATE Predictions by Meta Learner')
plt.xlabel('Individual Treatment Effect (ITE/CATE)')
plt.ylabel('# of Samples')
```



Estimate Average Treatment Effect (ATE)

```python
ate_s = learner_s.estimate_ate(X=X, treatment=treatment, y=y)
print(ate_s)
print('ATE estimate: {:.03f}'.format(ate_s[0][0]))
print('ATE lower bound: {:.03f}'.format(ate_s[1][0]))
print('ATE upper bound: {:.03f}'.format(ate_s[2][0]))
```

```
(array([0.6841716]), array([0.63612064]), array([0.73222256]))
ATE estimate: 0.684
ATE lower bound: 0.636
ATE upper bound: 0.732
```

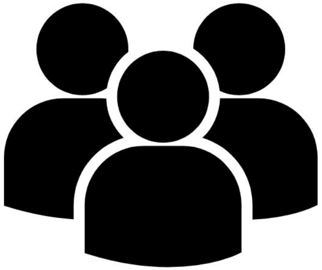
Meta-Learners example:
[examples/meta_learners_with_synthetic_data.ipynb]

CausalML | Value optimization

Knowing the CATE is often not sufficient to justify targeting populations if interventions are costly. Consider two scenarios:

1. The intervention costs more than the expected value of converting a customer
2. The customer is one that would convert anyway even in the absence of the intervention

These are some of the reasons why Causal ML contains modules for “value optimization”, ie for selecting populations for interventions in a way that optimizes ROI.



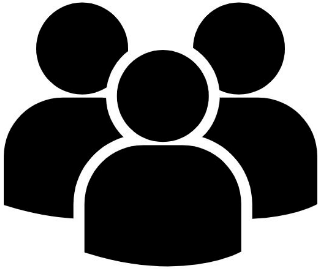
Probability of purchase under treatment: 0

Probability of purchase under control: 0

CATE: 0

Voucher: $5

Cost of targeting: 0



Probability of purchase under treatment: 1

Probability of purchase under control: 1

CATE: 0

Voucher: $5

Cost of targeting: -$5

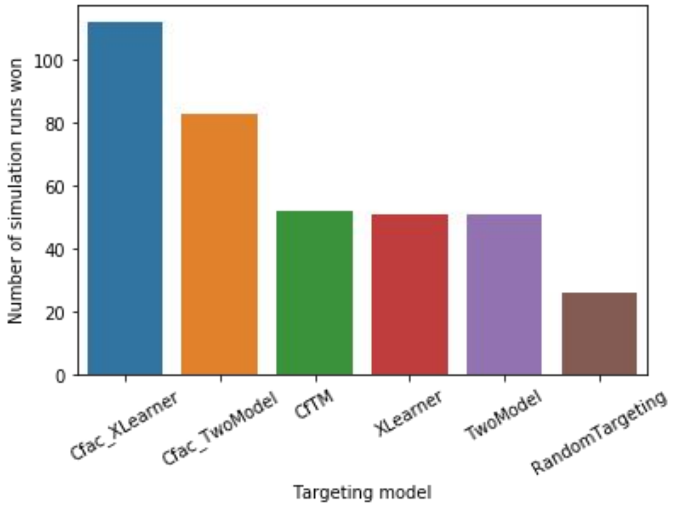
Zhao, Zhenyu, and Totte Harinen. "Uplift modeling for multiple treatments with cost optimization." 2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA). IEEE, 2019.
Ang Li, Judea Pearl. "Unit Selection Based on Counterfactual Logic." (2019).

CausalML | Value optimization example

```python
from causalml.optimize import CounterfactualValueEstimator
from causalml.optimize import get_treatment_costs, get_actual_value

# Run the counterfactual calculation with TwoModel prediction
cve = CounterfactualValueEstimator(treatment=df_test['treatment_group_key'],
                                   control_name='control',
                                   treatment_names=conditions[1:],
                                   y_proba=y_proba,
                                   cate=tm_pred,
                                   value=conversion_value_array[test_idx],
                                   conversion_cost=cc_array[test_idx],
                                   impression_cost=ic_array[test_idx])
cve_best_idx = cve.predict_best()
cve_best = [conditions[idx] for idx in cve_best_idx]
actual_is_cve_best = df.loc[test_idx, 'treatment_group_key'] == cve_best
cve_score = actual_value.loc[test_idx][actual_is_cve_best].mean()

# plot the result
plt.bar(model_labels, model_scores)
```



Counterfactual Value Optimization example:
examples/counterfactual_value_optimization.ipynb

CausalML | Overview

Modules

- Data Generation Process
- Feature Engineering
- Propensity Score Methods
- Sensitivity Analysis
- Causal Inference models
- Visualization

Uplift Trees

- KL Divergence
- Chi-Square
- Euclidean Distance
- Contextual Treatment Selection

Meta Learners

- R-Learner
- X-Learner
- T-Learner
- S-Learner
- DR-Learner
- TMLE

Neural Networks

- DragonNet
- CEVAE

Instrumental Variables

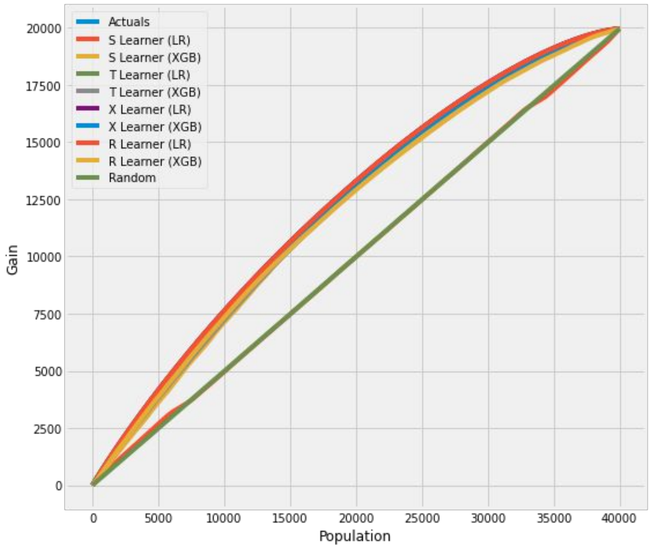
- 2SLS
- Doubly Robust IV

CausalML | Visualization: uplift curve

```python
from causalml.metrics import plot_gain, get_cumgain, auuc_score
from causalml.dataset import synthetic_data, simulate_nuisance_and_easy_treatment
from causalml.dataset import get_synthetic_preds_holdout, get_synthetic_auuc,

# Generate synthetic data for single simulation
train_preds, valid_preds = get_synthetic_preds_holdout(simulate_nuisance_and_easy_treatment,
                                                        n=50000,
                                                        valid_size=0.2)

# Cumulative Gain AUUC values for a Single Simulation of Validaiton Data
get_synthetic_auuc(train_preds)
```



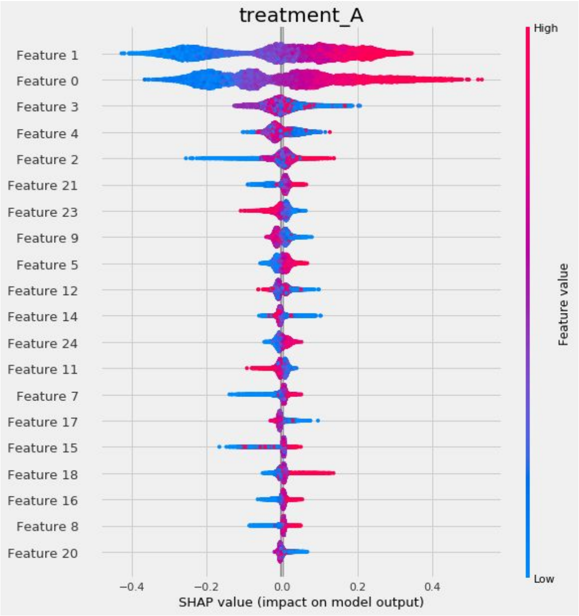
Model and feature interpretation example:
[examples/feature_interpretations_example.ipynb]()

CausalML | Visualization: SHAP

```
from causalml.inference.meta import BaseSRegressor

y, X, treatment, tau, b, e = synthetic_data(mode=1, n=n_samples, p=n_features, sigma=0.5)
w_multi = np.array(['treatment_A' if x==1 else 'control' for x in treatment])

slearner = BaseSRegressor(LGBMRegressor(), control_name='control')
slearner_tau = slearner.fit_predict(X, w_multi, y)
slearner.plot_shap_values(X=X, tau=slearner_tau, features=feature_names)
```



Model and feature interpretation example:
examples/feature_interpretations_example.ipynb

CausalML | Data generation

Modules

- Data Generation Process
- Feature Engineering
- Propensity Score Methods
- Sensitivity Analysis
- Causal Inference models
- Visualization

Classification dataset

```
from causalml.dataset import make_uplift_classification
# generate a synthetic dataset for classification uplift modeling problem
df, x_names = make_uplift_classification(n_samples=10000,
                                         treatment_name=['control', 'treatment1', 'treatment2'],
                                         y_name='conversion',
                                         n_classification_features=25,
                                         n_classification_informative=5,
                                         random_seed=2021)
```

Reproducibility

Flexible dataset customization

CausalML | Data generation

Modules

- Data Generation Process
- Feature Engineering
- Propensity Score Methods
- Sensitivity Analysis
- Causal Inference models
- Visualization

Classification dataset

```python
from causalml.dataset import make_uplift_classification
# generate a synthetic dataset for classification uplift modeling problem
df, x_names = make_uplift_classification(n_samples=10000,
                                         treatment_name=['control', 'treatment1', 'treatment2'],
                                         y_name='conversion',
                                         n_classification_features=25,
                                         n_classification_informative=5,
                                         random_seed=2021)
```

```
df.head()
```

	treatment_group_key	x1_informative	x2_informative	x3_informative	x4_informative	x5_informative	x6_irrelevant	x7_ir
0	control	0.796949	2.011955	1.010031	-0.082613	2.006405	0.853578	
1	treatment1	0.808755	0.027321	1.321761	-1.601562	0.701092	-0.943481	
2	control	0.092814	0.476592	-0.347588	0.319031	-0.523125	-0.142460	
3	control	0.769794	-2.382001	-0.353935	-1.643796	1.175124	1.302205	
4	control	0.753663	0.738282	-0.296090	-0.439421	0.067548	-0.178543	

```
df.treatment_group_key.value_counts()
```

```
treatment1    10000
control       10000
treatment2    10000
Name: treatment_group_key, dtype: int64
```

```
df.columns
```

```
Index(['treatment_group_key', 'x1_informative', 'x2_informative',
       'x3_informative', 'x4_informative', 'x5_informative', 'x6_irrelevant',
       'x7_irrelevant', 'x8_irrelevant', 'x9_irrelevant', 'x10_irrelevant',
       'x11_irrelevant', 'x12_irrelevant', 'x13_irrelevant', 'x14_irrelevant',
       'x15_irrelevant', 'x16_irrelevant', 'x17_irrelevant', 'x18_irrelevant',
       'x19_irrelevant', 'x20_irrelevant', 'x21_irrelevant', 'x22_irrelevant',
       'x23_irrelevant', 'x24_irrelevant', 'x25_irrelevant',
       'x26_uplift_increase', 'x27_uplift_increase', 'x28_increase_mix',
       'x29_uplift_increase', 'x30_uplift_increase', 'x31_increase_mix',
       'conversion', 'treatment_effect'],
      dtype='object')
```

CausalML | Data generation

Modules

- Data Generation Process
- Feature Engineering
- Propensity Score Methods
- Sensitivity Analysis
- Causal Inference models
- Visualization

Classification dataset

```
from causalml.dataset import make_uplift_classification
# generate a synthetic dataset for classification uplift modeling problem
df, x_names = make_uplift_classification(n_samples=10000,
                                         treatment_name=['control', 'treatment1', 'treatment2'],
                                         y_name='conversion',
                                         n_classification_features=25,
                                         n_classification_informative=5,
                                         random_seed=2021)
```

Regression dataset

```
from causalml.dataset.regression import synthetic_data

y, x, treatment, tau, b, e = synthetic_data(mode=1, sigma=0.5,
                                             n=NUM_SAMPLES,
                                             P=NUM_FEATURES)
```

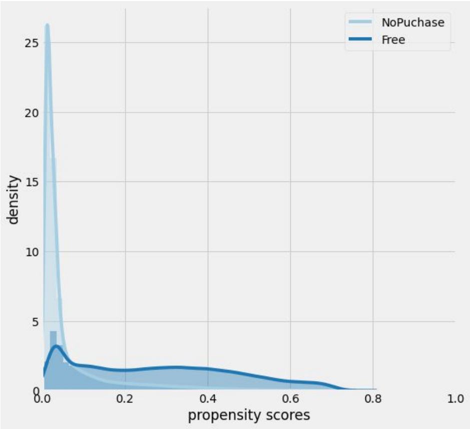
Support 5 modes of simulation [1][2]

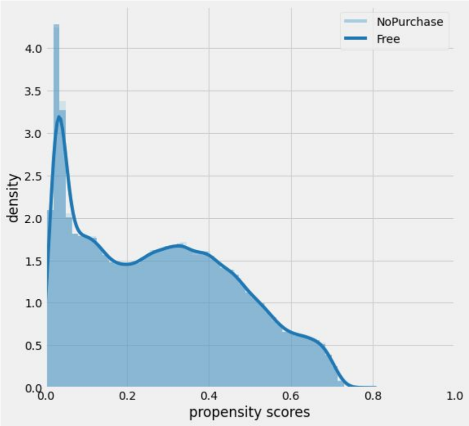
[1] Nie X. and Wager S. (2018) “Quasi-Oracle Estimation of Heterogeneous Treatment Effects”
[2] Louizos et al. (2018) "Causal Effect Inference with Deep Latent-Variable Models”

CausalML | Propensity score methods

Modules

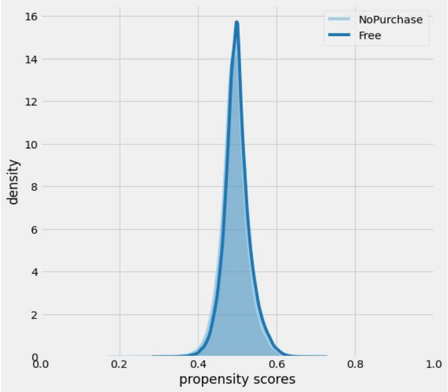
- Data Generation Process
- Feature Engineering
- **Propensity Score Methods**
- Sensitivity Analysis
- Causal Inference models
- Visualization





1.Pre and Post PSM

2. Re-Calibration



CausalML | Propensity score methods

Propensity Score Matching

```
from causalml.match import NearestNeighborMatch, MatchOptimizer, create_table_one
from causalml.propensity import ElasticNetPropensityModel

clf.train(df_train[PROPENSITY_FEATURES], df_train[TREATMENT_COL])
df['pihat'] = clf.predict(df[PROPENSITY_FEATURES])
matcher = NearestNeighborMatch(caliper=0.1, replace=True)
df_matched = matcher.match(data=df, treatment_col=TREATMENT_COL, score_cols=['pihat'])
```

Re-calibrate propensity

```
from causalml.features import OneHotEncoder

ohe = OneHotEncoder(min_obs=df.shape[0] * 0.01)
X = np.hstack([ ... .. , ohe.fit_transform(df_matched[ENCODING_COLS])])
p_model = ElasticNetPropensityModel()
df_matched['pihat_re'] = p_model.fit_predict(X, df_matched['conversion'])

# Print out SMD value for each feature after matching
create_table_one(df_matched, treatment_col= ... ,  features=MATCHING_COVARIATES)
```

SMD values

Variable	Control	Treatment	SMD
n	3656	3656	
pihat	0.07 (0.06)	0.07 (0.06)	0.0000
prob_features_11	0.39 (1.07)	0.36 (1.06)	-0.0295
prob_features_17	0.02 (0.81)	0.02 (0.84)	0.0021
prob_features_18	0.03 (0.82)	0.01 (0.82)	-0.0305
prob_features_19	-0.40 (0.79)	-0.35 (0.82)	0.0674
prob_features_20	0.56 (1.58)	0.41 (1.40)	-0.1040
prob_features_21	-0.07 (1.26)	-0.05 (1.05)	0.0162
prob_features_22	0.05 (0.94)	0.03 (0.86)	-0.0251
prob_features_23	0.61 (1.55)	0.46 (1.45)	-0.0941
prob_features_24	0.20 (1.08)	0.16 (1.06)	-0.0385

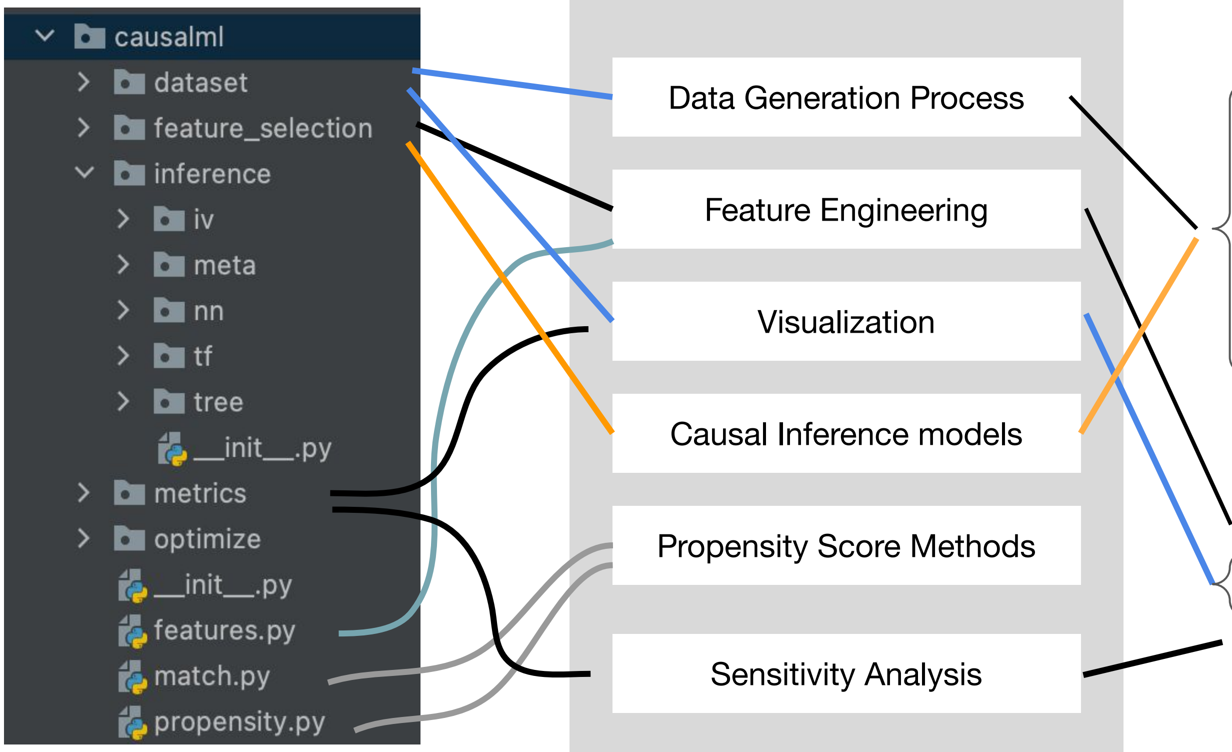
Propensity score matching: causalml/causalml/match.py
Sensitivity Analysis example: examples/sensitivity_example_with_synthetic_data.ipynb

Summary

CausalML | Overview

Package structure

Modules and Key algorithms



Notebook examples

https://github.com/uber/causalml/tree/master/examples

- meta_learners_with_synthetic_data
- meta_learners_with_synthetic_data_multiple_treatment
- uplift_trees_with_synthetic_data
- dr_learner_with_synthetic_data
- cevae_example
- dragonnet_example
- validation_with_tmle

optimize module

- counterfactual_value_optimization
- binary_policy_learner_example

- feature_selection
- uplift_tree_visualization
- feature_interpretations_example
- sensitivity_example_with_synthetic_data

Summary | Acknowledgement

Shoutout to all the contributors[1] of CausalML community!

Huigang Chen
Jeong-Yoon Lee
Jing Pan
Mike Yung
Paul Lo
Totte Harinen
Yifeng Wu
Zhenyu Zhao

Yuchen (@yluogit)
Manoj (@manojbalaji1)
Peter (@peterfoley)
Suraj (@surajiyer)
Harsh (@HarshCasper)
Fritz (@fritzo)
Tomasz (@TomaszZamacinski)
Georg (@waltherg)
Florian (@FlorianWilhelm)
Harry (@deeplaunch)
Katherine (@khof312)

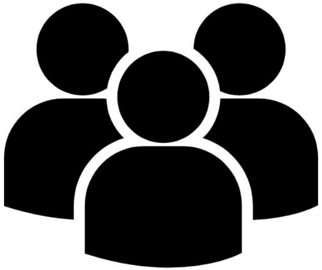
Steve (@steveyang90)
Vaclavbelak (@vaclavbelak)
Mario (@mwijaya3)
Christophe (@ccrndn)
Jannik (@jroessler)
Matthew (@maccam912)
Leo (@lleiou)
Mohamed (@ibraaaa)

[1] *List of contributors:* *https://github.com/uber/causalml/graphs/contributors*

Summary | Look forward to collaborating!

One-stop Shop

We **democratise uplift modeling methods** in academic paper and continue to add innovative in-house algorithms!



Dedicated support & Future collaborations

incubated for **long-term support** from Uber and community developers. **We welcome collaborations on package development and use cases!** https://github.com/uber/causalml#contributing

Industrial Applications

We aim to tackle **real-world large scale data and industrial applications**.

→ Let's dive into our Case Study section!

Questions & Comments

Appendix

Appendix | References

[1] Ahmed Alaa and Mihaela Schaar. Limits of estimating heterogeneous treatment effects: guidelines for practical algorithm design. In International Conference on Machine Learning, 129–138. 2018.
[2] Susan Athey and Guido Imbens. Recursive partitioning for heterogeneous causal effects. Proceedings of the National Academy of Sciences, 113(27):7353–7360, 2016.
[3] Susan Athey, Julie Tibshirani, Stefan Wager, and others. Generalized random forests. The Annals of Statistics, 47(2):1148–1178, 2019.
[4] Susan Athey and Stefan Wager. Efficient policy learning. arXiv preprint arXiv:1702.02896, 2017.
[5] Pierre Gutierrez and Jean-Yves Gerardy. Causal inference and uplift modeling a review of the literature. JMLR: Workshop and Conference Proceedings 67, 2016.
[6] Jason Hartford, Greg Lewis, Kevin Leyton-Brown, and Matt Taddy. Deep iv: a flexible approach for counterfactual prediction. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, 1414–1423. JMLR. org, 2017.
[7] Guido W Imbens and Jeffrey M Wooldridge. Recent developments in the econometrics of program evaluation. Journal of economic literature, 47(1):5–86, 2009.
[8] Sören R Künzel, Jasjeet S Sekhon, Peter J Bickel, and Bin Yu. Metalearners for estimating heterogeneous treatment effects using machine learning. Proceedings of the National Academy of Sciences, 116(10):4156–4165, 2019.

Appendix | References

[9] Xinkun Nie and Stefan Wager. Quasi-oracle estimation of heterogeneous treatment effects. arXiv preprint arXiv:1712.04912, 2017.

[10] Miruna Oprescu, Vasilis Syrgkanis, and Zhiwei Steven Wu. Orthogonal random forest for heterogeneous treatment effect estimation. CoRR, 2018. URL: http://arxiv.org/abs/1806.03467, arXiv:1806.03467.

[11] Piotr Rzepakowski and Szymon Jaroszewicz. Decision trees for uplift modeling with single and multiple treatments. Knowl. Inf. Syst., 32(2):303–327, August 2012.

[12] Yan Zhao, Xiao Fang, and David Simchi-Levi. Uplift modeling with multiple treatments and general response types. In Proceedings of the 2017 SIAM International Conference on Data Mining, 588–596. SIAM, 2017.

[13] Hahn, P. Richard, Jared S. Murray, and Carlos M. Carvalho. "Bayesian Regression Tree Models for Causal Inference: Regularization." Confounding, and Heterogeneous Effects 1706 (2017).

[14] Chen, Huigang, Totte Harinen, Jeong-Yoon Lee, Mike Yung, and Zhenyu Zhao. "Causalml: Python package for causal machine learning." arXiv preprint arXiv:2002.11631 (2020).

[15] Zhao, Zhenyu, Yumin Zhang, Totte Harinen, and Mike Yung. "Feature Selection Methods for Uplift Modeling." arXiv preprint arXiv:2005.03447 (2020).

[16] Zhao, Zhenyu, and Totte Harinen. "Uplift modeling for multiple treatments with cost optimization." In 2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 422-431. IEEE, 2019.