Microsoft Research

EconML: A Machine Learning Library for Estimating Heterogeneous Treatment Effects

Miruna Oprescu   Vasilis Syrgkanis   Keith Battocchi   Maggie Hei   Greg Lewis

github.com/Microsoft/EconML

ALICE
Automated Learning and Intelligence for Causation and Economics

Why EconML?

- Implements recent techniques that tackle heterogeneous treatment effect estimation from observational data via **machine learning-based approaches**
- Incorporates **techniques form recent academic works** (e.g. Double Machine Learning, Causal Forests, Deep Instrumental Variables, Meta-learners[1], etc.) under a common API
- Empowers researchers/data scientists/decision-makers to perform **causal analysis without extensive Economics training**
- Open source go-to causal analysis toolkit built on standard machine learning packages that provides built-in **cross-validation, inference, interpretability**, all in one place

[1]For a complete list of references see econml.azurewebsites.net/spec/references.html

Setup and API Design

T — treatment policy, Y — outcome of intervention
X — features that capture heterogeneity (optional)
W — controls (optional)
Z — instruments (optional)

Potential Outcomes Formulation:

$Y(t) \rightarrow$ potential outcome

- CATE:
  $$\tau(t_0, t_1, x) = \mathbb{E}[Y(t_1) - Y(t_0) | X = x]$$
- Marginal CATE:
  $$\partial\tau(t, x) = \mathbb{E}[\nabla_t Y(t) | X = x]$$
- Counterfactual prediction:
  $$\mu(t, x) = \mathbb{E}[Y(t) | X = x]$$

Structural Equations Formulation:

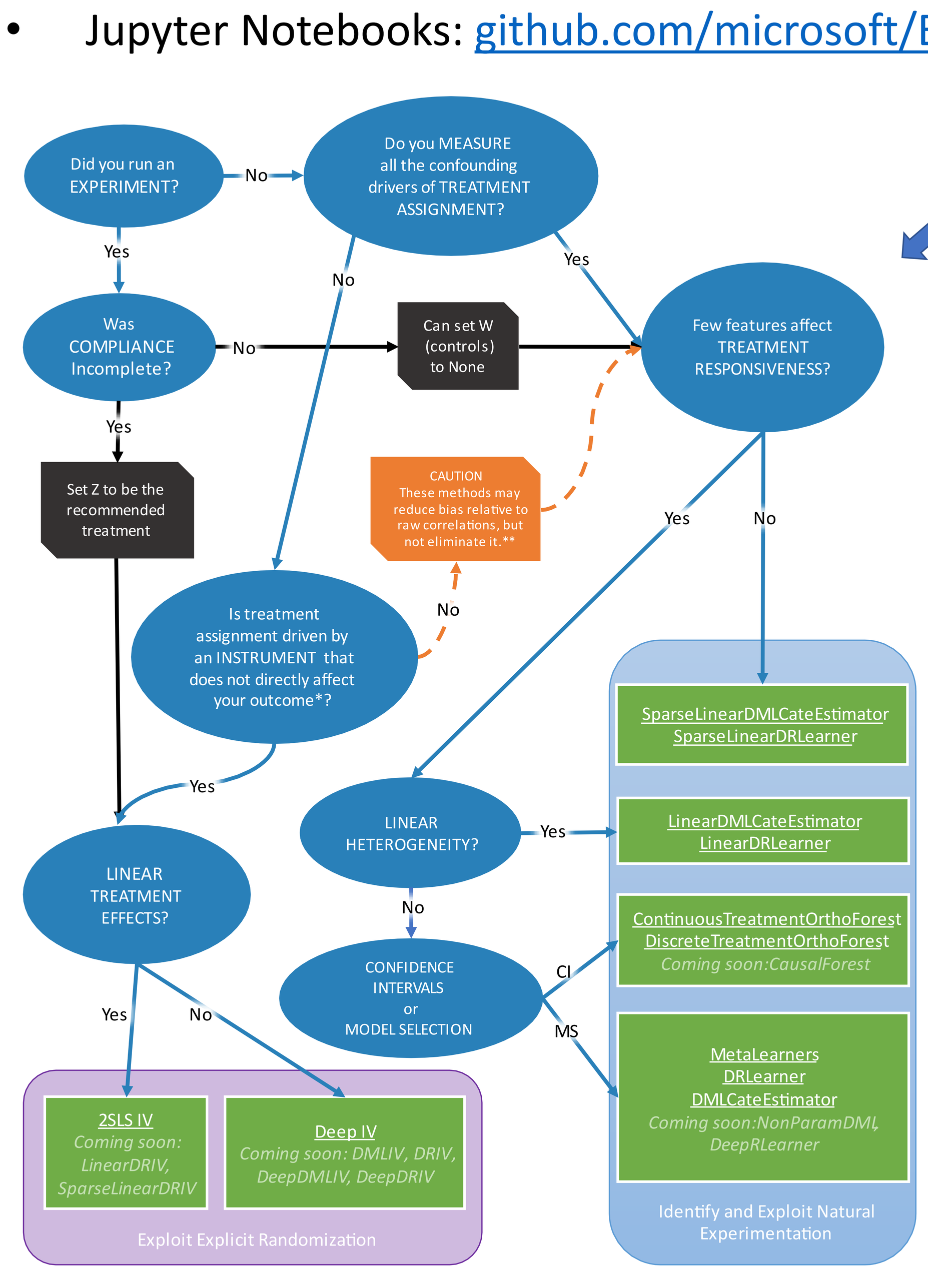
- Model:
  $$Y = g(T, X, W, \epsilon); \;\; T = f(X, W, Z, \eta)$$
- CATE:
  $$\tau(t_0, t_1, x) = \mathbb{E}[g(t_1, X, W, \epsilon) - g(t_0, X, W, \epsilon) | X = x]$$
- Marginal CATE:
  $$\partial\tau(t, x) = E[\nabla_t g(t, X, W, \epsilon) | X = x]$$

```
class BaseCateEstimator:

    def fit(self, Y, T, X=None, W=None, Z=None, inference=None):
        ''' Estimates the counterfactual model from data, i.e. estimates functions
        τ(·,·, ·)}, ∂τ(·, ·) and μ(·, ·)
        inference → Method for performing inference.  All estimators support 'bootstrap'
                    some support other methods as well.'''

    def effect(self, X=None, *, T0, T1):
        ''' Calculates the heterogeneous treatment effect τ(·, ·, ·) between two
        treatment points conditional on a vector of features X '''

    def marginal_effect(self, T, X=None):
        ''' Calculates the heterogeneous marginal effect ∂τ(·, ·) around a base
        treatment point conditional on a vector of features X '''

    def effect_interval(self, X=None, *, T0=0, T1=1, alpha=0.1):
        ''' Confidence intervals for the quantities τ(·, ·, ·) produced by the model.
        `alpha` corresponds to (1 - alpha) level of confidence '''

    def marginal_effect_interval(self, T, X=None, *, alpha=0.1):
        ''' Confidence intervals for the quantities ∂τ(·, ·) produced by the model.'''
```

Fig. 1: Snapshot of the common CATE API implemented in Python

Try it Out!

- Python: pip install econml
- GitHub: github.com/microsoft/EconML
- Documentation: econml.azurewebsites.net
- Jupyter Notebooks: github.com/microsoft/EconML/tree/master/notebooks



Familiarize yourself with the various EconML estimators and their properties!

Estimator	Treatment Type	Instrument?	Confidence Intervals?	Linear Treatment?	Linear Heterogeneity	Multiple Outcomes?	Multiple Treatments?	High-diml' Features?
NonparametricTwoStageLeastSquares	Any	✓		✓	Assumed	✓	✓	
DeepIVEstimator	Any	✓				✓	✓	
SparseLinearDMLCateEstimator	Any		✓	✓	Assumed	✓	✓	✓
SparseLinearDRLearner	Categorical		✓		Projected		✓	✓
LinearDMLCateEstimator	Any		✓	✓	Assumed	✓	✓	
LinearDRLearner	Categorical		✓		Projected		✓	
ForestDMLCateEstimator	1-d/Binary		✓	✓		✓		✓
ForestDRLearner	Categorical		✓			✓	✓	✓
ContinuousTreatmentOrthoForest	Continuous		✓	✓			✓	✓
DiscreteTreatmentOrthoForest	Categorical		✓				✓	✓
metalearners	Categorical						✓	✓
DRLearner	Categorical						✓	✓
DMLCateEstimator	Any			✓	Assumed	✓	✓	✓

Usage Examples

Example with Built-in Cross-Validation

- We estimate the effect of orange juice price (*T*) on demand (*Y*). The data contains several features *W*, but we want to learn the elasticity of demand as a function of income alone (*X*)
- We apply the Double Machine Learning (DML) technique with a polynomial effect

```
from econml.dml import LinearDMLCateEstimator

# Parameter sweep for cross-validated random forest
rf_params = {'max_depth' : [5, 10, 15]}
# Cate estimator
cate_est = LinearDMLCateEstimator(
            model_y = GridSearchCV(RandomForestRegressor(), rf_params), # Built-in cross-validation
            model_t = GridSearchCV(RandomForestRegressor(), rf_params), # Built-in cross-validation
            featurizer = PolynomialFeatures(degree=3)
          )
# Fit estimator with inference and calculate treatment effects
cate_est.fit(Y, T, X, W, inference= 'statsmodels' )
te_pred = cate_est.effect(X_test)
```

Inference

The EconML estimators support one or more of the following inference methods:

- Bootstrap (inference='bootstrap')
- OLS (inference='statsmodels')
- Debiased Lasso (inference='debiasedlasso')
- Subsample Honest Forest (Bootstrap of Little Bags, inference='blb')

```
# Building confidence intervals
lower, upper = cate_est.effect_interval(X_test, alpha=0.02)
```

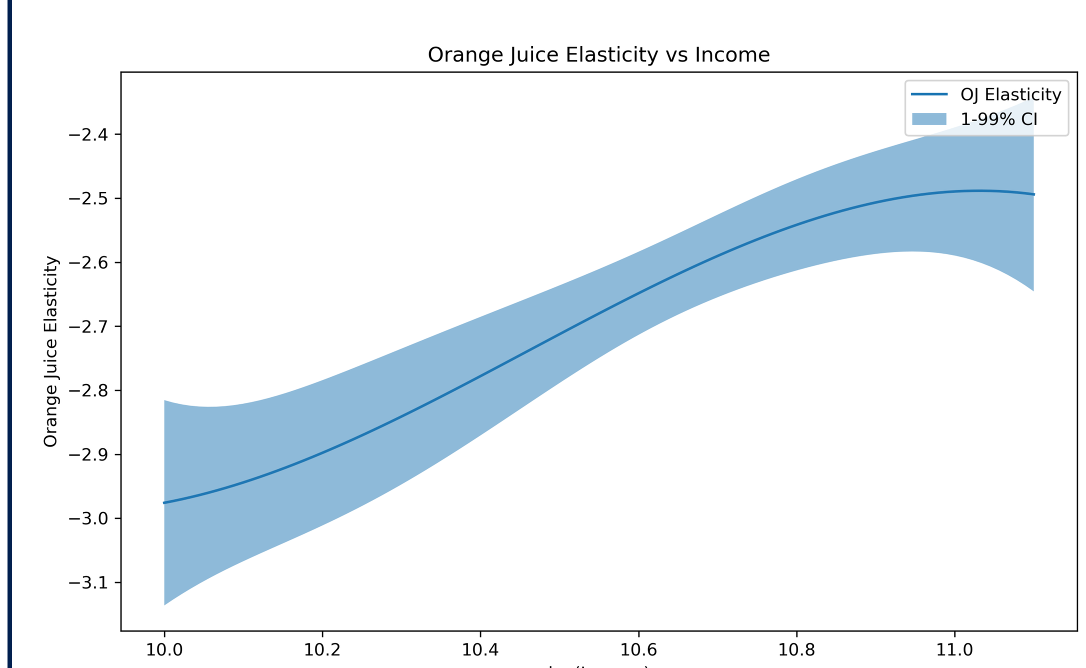


Fig. 2: Linear DML estimates for the effect of orange juice price on demand by income level. The shaded region depicts the 1-99% confidence interval. The results unveil the natural phenomenon that lower income consumers are more price-sensitive.

Interpretability

The EconML interpretability toolkit offers:

- Tools for interpreting effects heterogeneity and treatment policies
- Integration with Python visualization libraries such as Graphviz and SHAP

```
from econml.cate_interpreter import SingleTreeCateInterpreter

intrp = SingleTreeCateInterpreter(include_model_uncertainty=True, max_depth=2, min_samples_leaf=10)
# We interpret the CATE models behavior on the distribution of heterogeneity features
intrp.interpret(est, X_test)
# We directly render the tree using the graphviz python library
intrp.render(out_file='oj_cate_tree', format='png', view=True, feature_names=['log(Income)'])
```

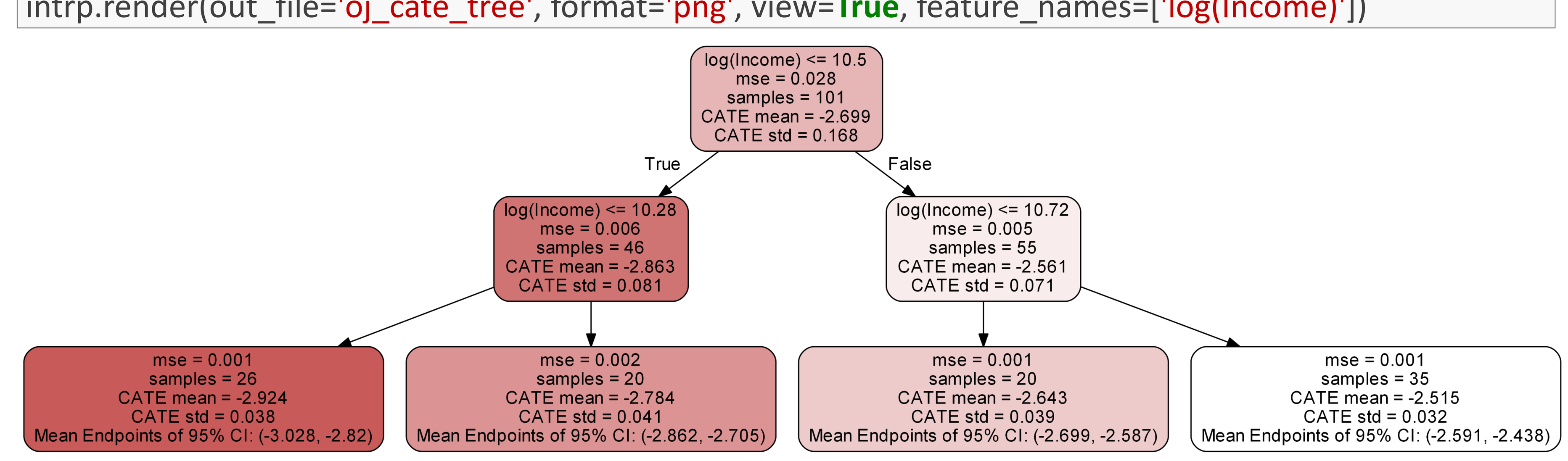


Fig. 3: Tree-based interpretation of orange juice elasticity estimates