Amazon Case Study: Detailed EconML Method Explanations

# 1. LinearDML

## Why Use This Method?

Use LinearDML when you want to estimate how a continuous treatment like price affects an outcome like purchase quantity, controlling for confounders using ML. Best for interpretable marginal effects with confidence intervals.

## Key Assumptions

• Unconfoundedness: T ⟂ ε | X, W  
• Linear effect: Y = θ(X) · T + f(X, W) + ε  
• Orthogonality: residuals uncorrelated with errors.

## Model Structure

Predict Y and T using ML (g(X,W), f(X,W)), regress outcome residuals on treatment residuals to get θ(X).

## How to Implement

1. Define variables (Y, T, X, W).  
2. Use ML to model Y ~ X, W and T ~ X, W.  
3. Regress residuals to estimate effect.  
4. Predict with est.const\_marginal\_effect(X\_test).

## Expected Results

Returns marginal treatment effect by subgroup.

## How to Interpret Results

θ(X\_test) = -0.9 → $1 price drop increases demand by 0.9 units.

## What Control Variables to Include (W)

• Time factors: month, day, holiday  
• Purchase history: prior\_purchases, avg\_spend  
• Platform: device\_type, channel  
• Inventory: stock\_level  
• Region: geographic zone, urban/rural  
• Marketing: ad\_seen, email\_clicked

# 2. DRLearner

## Why Use This Method?

Use DRLearner for robustness to model misspecification. It's consistent if either the outcome or the treatment model is correctly specified.

## Key Assumptions

• Either outcome model or propensity model must be correct.  
• No hidden confounding.  
• Can handle flexible ML models.

## Model Structure

Estimate g\_t(X,W) and p\_t(X,W), compute pseudo-outcomes, fit final model for CATE.

## How to Implement

1. Estimate outcome and propensity models.  
2. Construct doubly robust scores.  
3. Fit final CATE estimator.

## Expected Results

CATE estimate is reliable even if one nuisance model is misspecified.

## How to Interpret Results

θ(X) = -0.8 → $1 discount raises demand by 0.8 units for that X.

## What Control Variables to Include (W)

• Time factors: month, day, holiday  
• Purchase history: prior\_purchases, avg\_spend  
• Platform: device\_type, channel  
• Inventory: stock\_level  
• Region: geographic zone, urban/rural  
• Marketing: ad\_seen, email\_clicked

# 3. CausalForestDML

## Why Use This Method?

Use CausalForestDML for flexible, non-parametric CATE estimation. Great for segment-specific effects.

## Key Assumptions

• Same as DML: assumes confounders are observed.  
• Non-parametric model for treatment effect.

## Model Structure

Random forest regression with orthogonalized inputs.

## How to Implement

1. Fit models for Y and T with ML.  
2. Apply causal forest to estimate CATE.

## Expected Results

Gives individual-level effects and confidence intervals.

## How to Interpret Results

Can show which segments (e.g., Prime users) are most price sensitive.

## What Control Variables to Include (W)

• Time factors: month, day, holiday  
• Purchase history: prior\_purchases, avg\_spend  
• Platform: device\_type, channel  
• Inventory: stock\_level  
• Region: geographic zone, urban/rural  
• Marketing: ad\_seen, email\_clicked

# 4. DMLIV

## Why Use This Method?

Use DMLIV when the treatment is endogenous (e.g., algorithmic price setting) and you have a valid instrument.

## Key Assumptions

• Z affects T but not Y directly.  
• Exclusion restriction: Z ⟂ ε | X, W.  
• Instrument relevance: T = h(Z,X,W) + η.

## Model Structure

Stage 1: model T with Z.  
Stage 2: predict Y from T.  
Estimate treatment effect using residualized versions.

## How to Implement

1. Define instrument Z.  
2. Fit models for Y, T, and Z.  
3. Estimate effect with DMLIV.  
4. Predict using est.effect(X\_test).

## Expected Results

Estimates causal effect of treatment using instruments.

## How to Interpret Results

θ(X) = -1.0 → Validated elasticity estimate after removing endogeneity.

## What Control Variables to Include (W)

• Time factors: month, day, holiday  
• Purchase history: prior\_purchases, avg\_spend  
• Platform: device\_type, channel  
• Inventory: stock\_level  
• Region: geographic zone, urban/rural  
• Marketing: ad\_seen, email\_clicked

# 5. DynamicDML

## Why Use This Method?

Use DynamicDML when treatments and outcomes evolve over time (e.g., sequential discounts).

## Key Assumptions

• Sequential ignorability  
• Markovian structure  
• Treatment policy evolves over time.

## Model Structure

Uses past states and actions to predict present outcomes. Models time-dependent effect.

## How to Implement

1. Structure data as panel.  
2. Group by customer.  
3. Fit dynamic DML estimator.

## Expected Results

Estimates how ongoing promotions affect later purchases.

## How to Interpret Results

θ(X\_t) = -0.7 → discounts now lead to higher sales later.

## What Control Variables to Include (W)

• Time factors: month, day, holiday  
• Purchase history: prior\_purchases, avg\_spend  
• Platform: device\_type, channel  
• Inventory: stock\_level  
• Region: geographic zone, urban/rural  
• Marketing: ad\_seen, email\_clicked

# 6. Meta Learners

## Why Use This Method?

Use when you want quick, interpretable models or start exploring treatment heterogeneity.

## Key Assumptions

• Different methods for different assumptions:  
• T: Separate models for treated/control.  
• S: Single model with treatment flag.  
• X: Blends both and reweights.  
• DomainAdapt: adjusts for population imbalance.

## Model Structure

Basic model estimation framework:  
T-learners: f1(X), f0(X)  
S-learners: f(X, T)  
X-learners: weighted combination.

## How to Implement

1. Split data.  
2. Estimate outcome models.  
3. Subtract to get CATE.

## Expected Results

Estimates subgroup treatment effects. Easy to debug.

## How to Interpret Results

θ(X) = -0.5 → $1 price drop causes 0.5 unit increase for given group.

## What Control Variables to Include (W)

• Time factors: month, day, holiday  
• Purchase history: prior\_purchases, avg\_spend  
• Platform: device\_type, channel  
• Inventory: stock\_level  
• Region: geographic zone, urban/rural  
• Marketing: ad\_seen, email\_clicked

# 7. OrthoForest

## Why Use This Method?

Use OrthoForest when data is high-dimensional and you want flexible yet statistically valid CATE estimates.

## Key Assumptions

• Smoothness assumption on θ(X).  
• Local moment conditions hold.  
• Orthogonalized residuals used with forest.

## Model Structure

Combines forest regression with DML orthogonalization.

## How to Implement

1. Estimate residuals from Y and T.  
2. Fit local forest models.  
3. Aggregate to estimate CATE.

## Expected Results

Accurate CATE estimation in large, complex datasets.

## How to Interpret Results

Visualizes heterogeneity with statistical confidence.

## What Control Variables to Include (W)

• Time factors: month, day, holiday  
• Purchase history: prior\_purchases, avg\_spend  
• Platform: device\_type, channel  
• Inventory: stock\_level  
• Region: geographic zone, urban/rural  
• Marketing: ad\_seen, email\_clicked

# 8. DeepIV

## Why Use This Method?

Use DeepIV when you want to estimate causal effects under endogeneity using deep neural networks.

## Key Assumptions

• Requires strong instruments.  
• Flexible ML for complex, nonlinear models.

## Model Structure

Uses deep nets for both stages: T ~ Z and Y ~ T\_hat.

## How to Implement

1. Train deep network T ~ Z, X.  
2. Train second network Y ~ predicted T.  
3. Predict effect.

## Expected Results

Estimates highly nonlinear CATE functions.

## How to Interpret Results

θ(X) = -1.3 → Deep price elasticity learned from complex relationships.

## What Control Variables to Include (W)

• Time factors: month, day, holiday  
• Purchase history: prior\_purchases, avg\_spend  
• Platform: device\_type, channel  
• Inventory: stock\_level  
• Region: geographic zone, urban/rural  
• Marketing: ad\_seen, email\_clicked

# 9. SieveTSLS

## Why Use This Method?

Use for interpretable nonparametric IV estimation using basis expansions (e.g., polynomials).

## Key Assumptions

• Basis expansion needed (Hermite, polynomial).  
• Valid instrument required.  
• 2SLS setup.

## Model Structure

1. Use basis expansion on Z, X, T.  
2. Fit linear IV estimator on basis.

## How to Implement

1. Define basis features.  
2. Fit 2SLS model.  
3. Interpret coefficients.

## Expected Results

Provides interpretable IV estimates.

## How to Interpret Results

θ = -0.6 → smooth nonlinear effect of price.

## What Control Variables to Include (W)

• Time factors: month, day, holiday  
• Purchase history: prior\_purchases, avg\_spend  
• Platform: device\_type, channel  
• Inventory: stock\_level  
• Region: geographic zone, urban/rural  
• Marketing: ad\_seen, email\_clicked

# 10. FederatedEstimator

## Why Use This Method?

Use when data is split across locations/devices and cannot be pooled due to privacy/storage limits.

## Key Assumptions

• Each node estimates locally.  
• Central server aggregates effects.  
• Assumes similar model specs.

## Model Structure

Train multiple estimators separately. Combine them.

## How to Implement

1. Fit DML/DRLearner per data region.  
2. Aggregate with FederatedEstimator.  
3. Apply globally.

## Expected Results

Builds global effect model from distributed data.

## How to Interpret Results

Allows Amazon to pool insights without moving raw data.

## What Control Variables to Include (W)

• Time factors: month, day, holiday  
• Purchase history: prior\_purchases, avg\_spend  
• Platform: device\_type, channel  
• Inventory: stock\_level  
• Region: geographic zone, urban/rural  
• Marketing: ad\_seen, email\_clicked

Perfect — below is the full structured explanation for all 10 EconML methods using the same format you requested, starting with **LinearDML**:

**🔍 Method: LinearDML**

Great question! **LinearDML** is used when you're estimating the **marginal effect of a continuous treatment** (like price) on an outcome (like purchases), while adjusting for **confounding variables** using machine learning.

It works well when:

* You want **interpretable treatment effects**,
* Your treatment effect is assumed to be **linear**, and
* You have many covariates to control for.

**✅ When to Use LinearDML at Amazon**

Suppose Amazon offers a **$10 discount on Echo Dots**, and you want to know:

“What is the causal effect of this price cut on units sold, adjusting for user behavior, seasonality, and Prime status?”

Use LinearDML to:

* Adjust for confounders with ML,
* Estimate **causal, marginal price effects**,
* Report interpretable results to stakeholders.

**🧠 Key Assumptions**

* **Unconfoundedness**: T⊥ε∣X,WT \perp \varepsilon \mid X, W
* **Linearity**: The effect of treatment TT on YY is linear.
* **Orthogonality**: Residuals from ML models are uncorrelated.

**🔧 Model Steps**

1. Predict Y∼X,WY \sim X, W using ML.
2. Predict T∼X,WT \sim X, W using ML.
3. Residualize YY and TT, then regress Yres∼TresY\_{res} \sim T\_{res}.

**⚙️ Example Implementation**

from econml.dml import LinearDML

from sklearn.ensemble import GradientBoostingRegressor

est = LinearDML(model\_y=GradientBoostingRegressor(),

model\_t=GradientBoostingRegressor(),

discrete\_treatment=False)

est.fit(Y, T, X=X, W=W)

theta = est.const\_marginal\_effect(X\_test)

**📈 Interpretation**

* If theta = -0.9, a $1 decrease in price is expected to **increase purchases by 0.9 units** for customers like X\_test.

**🛍 Amazon Case Study: Echo Dot Prime Promotion**

You want to measure:

“What is the effect of a $10 discount for Prime members in New York during Prime Day?”

LinearDML gives you a **precise, interpretable elasticity** that accounts for seasonality, loyalty, and regional behavior.

**✅ Control Variables (W): What to Include**

**🧩 1. Seasonality & Timing**

* month, holiday\_flag, day\_of\_week

**🧩 2. User Purchase History**

* prior\_purchases, avg\_spend, signup\_month

**🧩 3. Platform Context**

* device\_type, login\_status

**🧩 4. Inventory & Exposure**

* stock\_level, email\_opened, ad\_exposure

**✅ Summary Table**

| **Control Variable** | **Why Include?** |
| --- | --- |
| month, holiday\_flag | Capture sales variation due to time |
| prior\_purchases | Buyer type and loyalty indicators |
| device\_type | Mobile vs desktop may influence browsing and buying |
| stock\_level | Scarcity affects urgency and demand |

**🚫 What Not to Include**

Avoid:

* **Post-treatment behavior**, e.g., clicked\_after\_discount, time\_on\_checkout

**🧪 Code Example for W**

W = df[[

'month', 'holiday\_flag', 'prior\_purchases',

'avg\_spend', 'device\_type', 'stock\_level',

'email\_opened', 'region'

]].values

Great! I’ll now prepare the remaining 9 methods in the same structured format as LinearDML and present them here, one by one. Next up:

## 🔍 Method: DRLearner

Great question! **DRLearner** stands for **Doubly Robust Learner**, and it's used to estimate **Conditional Average Treatment Effects (CATE)** in observational data. Its doubly robust nature means it’s **valid if either the model for treatment or the model for outcome is correct**.

This makes it very useful when you're unsure which part of your model might be mis-specified.

## ✅ When to Use DRLearner at Amazon

Let’s say you want to know the **impact of showing a homepage banner with a discount** on actual purchases:

* But the assignment of the banner is not random.
* And there may be many covariates influencing both exposure and outcomes.

DRLearner allows you to adjust for this bias using ML.

## 🧠 Key Assumptions

* **No unobserved confounding**: T⊥ε∣X,WT \perp \varepsilon \mid X, W
* **Model robustness**: Valid if **either** the treatment model or outcome model is correct.

## 🔧 Model Steps

1. Predict potential outcomes for both treated and control groups.
2. Predict treatment probabilities (propensity scores).
3. Form doubly robust pseudo-outcomes and estimate treatment effect using ML.

## ⚙️ Example Implementation

from econml.dr import DRLearner

from sklearn.ensemble import RandomForestRegressor

est = DRLearner(model\_regression=RandomForestRegressor(),

model\_propensity=RandomForestRegressor())

est.fit(Y, T, X=X, W=W)

tau = est.effect(X\_test)

## 📈 Interpretation

* If tau = -1.2 for a given user profile, showing the discount banner **decreases purchases by 1.2 units** for users like them — possibly due to dilution effects or over-targeting.

## 🛍 Amazon Case Study: Homepage Promotion Analysis

“Does showing a discount banner boost purchases for new users vs. loyal customers?”

DRLearner helps analyze this by adjusting for selection bias in banner exposure and isolating the effect.

## ✅ Control Variables (W): What to Include

### 🧩 1. ****User Behavior History****

* prior\_clicks, avg\_basket\_value, last\_login\_days\_ago

### 🧩 2. ****Marketing Context****

* saw\_previous\_campaign, email\_engaged

### 🧩 3. ****Platform Experience****

* device\_type, time\_on\_site, search\_query\_count

### ✅ Summary Table

| **Control Variable** | **Why Include?** |
| --- | --- |
| prior\_clicks | Users who click more may be shown banners more |
| device\_type | Device affects ad layout and shopping behavior |
| email\_engaged | High correlation with marketing exposure |

## 🚫 What Not to Include

Avoid:

* Banner reaction behavior, like clicked\_discount\_banner

## 🧪 Code Example for W

W = df[[

'prior\_clicks', 'avg\_basket\_value', 'last\_login\_days\_ago',

'email\_engaged', 'device\_type', 'search\_query\_count'

]].values

Perfect! Here's the structured case-based explanation for:

## 🔍 Method: CausalForestDML

Great question! **CausalForestDML** is a nonparametric method that uses **random forests** to estimate **heterogeneous treatment effects**. It is designed to handle **complex, nonlinear interactions** between covariates and treatment effects.

It shines when you believe **different users or products respond differently to the same treatment** (like a discount), and you want to model that variation.

## ✅ When to Use CausalForestDML at Amazon

Let’s say Amazon offers a **$5 discount on Fire Tablets** across different customers. You want to know:

“Which types of customers respond best to the discount?”

Use CausalForestDML when:

* You suspect treatment effects vary (e.g., new users respond differently than loyal ones),
* You want flexible models without specifying functional forms,
* You need **confidence intervals for treatment effects**.

## 🧠 Key Assumptions

* **Unconfoundedness**: T⊥ε∣X,WT \perp \varepsilon \mid X, W
* **Overlap**: All units have non-zero probability of receiving each treatment.
* **Orthogonality**: ML models residualize Y and T properly.

## 🔧 Model Steps

1. Estimate residuals: Y~=Y−Y^(X,W)\tilde{Y} = Y - \hat{Y}(X, W), T~=T−T^(X,W)\tilde{T} = T - \hat{T}(X, W)
2. Fit a **causal forest** on Y~,T~\tilde{Y}, \tilde{T}
3. Estimate τ(X)\tau(X) and confidence intervals

## ⚙️ Example Implementation

from econml.dml import CausalForestDML

from sklearn.ensemble import RandomForestRegressor

est = CausalForestDML(model\_t=RandomForestRegressor(),

model\_y=RandomForestRegressor(),

discrete\_treatment=False)

est.fit(Y, T, X=X, W=W)

tau = est.effect(X\_test)

intervals = est.effect\_interval(X\_test)

## 📈 Interpretation

* If tau = -2.3 for a specific customer profile, this customer will likely buy **2.3 more units** under the discount.
* Confidence intervals help evaluate **uncertainty** in the estimate.

## 🛍 Amazon Case Study: Targeted Tablet Promotion

“Which customer segments benefit most from a $5 price drop on Fire Tablets?”

Use CausalForestDML to:

* Discover segments with high causal uplift,
* Inform personalized pricing or retargeting campaigns.

## ✅ Control Variables (W): What to Include

### 🧩 1. ****User Engagement****

* days\_since\_last\_purchase, pages\_viewed, cart\_additions

### 🧩 2. ****Demographics or Segments****

* account\_age, region, device\_type, customer\_segment

### 🧩 3. ****Product Context****

* inventory\_days, previous\_discount\_received, time\_spent\_on\_product\_page

### ✅ Summary Table

| **Control Variable** | **Why Include?** |
| --- | --- |
| pages\_viewed, cart\_additions | Influence both price exposure and conversion |
| region, device\_type | Capture geographic/tech context |
| previous\_discount\_received | Adjusts for promotion familiarity bias |

## 🚫 What Not to Include

Avoid:

* Post-discount actions like time\_spent\_on\_checkout\_after\_discount

## 🧪 Code Example for W

W = df[[

'days\_since\_last\_purchase', 'pages\_viewed', 'cart\_additions',

'account\_age', 'device\_type', 'region', 'previous\_discount\_received'

]].values

Great! Here's the structured explanation for:

## 🔍 Method: DMLIV (Double Machine Learning with Instrumental Variables)

Great question! **DMLIV** is designed to handle **endogeneity** — situations where your treatment (e.g., price) is **not randomly assigned** and may be **correlated with unobserved factors** that also affect your outcome (e.g., purchases).

This method uses **instrumental variables (IVs)** and ML to consistently estimate treatment effects, especially when there's **selection bias** or **strategic pricing**.

## ✅ When to Use DMLIV at Amazon

Let’s say you want to evaluate the **impact of algorithmically set prices on product demand**, but:

* Pricing depends on **past sales, browsing, or promotions**, which are also correlated with demand.
* You have a **randomized ad campaign or A/B price test** that can act as an instrument.

DMLIV allows you to **recover the causal price effect** even when treatment assignment is biased.

## 🧠 Key Assumptions

* **Instrument Relevance**: The instrument ZZ must predict treatment TT.  
  Cov(Z,T)≠0\text{Cov}(Z, T) \neq 0
* **Instrument Exogeneity**: The instrument is uncorrelated with the error term in the outcome equation.  
  Z⊥ε∣X,WZ \perp \varepsilon \mid X, W
* **Exclusion Restriction**: The instrument only affects YY through TT

## 🔧 Model Steps

1. Predict T∼Z,X,WT \sim Z, X, W: instrument → treatment
2. Predict Y∼T,X,WY \sim T, X, W
3. Estimate CATE: θ(X)=∂Y∂T\theta(X) = \frac{\partial Y}{\partial T} corrected using residuals

## ⚙️ Example Implementation

from econml.iv.dml import DMLIV

from sklearn.ensemble import GradientBoostingRegressor

est = DMLIV(model\_y=GradientBoostingRegressor(),

model\_t=GradientBoostingRegressor(),

model\_z=GradientBoostingRegressor(),

discrete\_treatment=False)

est.fit(Y, T, Z=Z, X=X, W=W)

theta = est.const\_marginal\_effect(X\_test)

## 📈 Interpretation

* If theta = -1.5, then a $1 price decrease **causes** 1.5 more units to be purchased, accounting for endogeneity.

## 🛍 Amazon Case Study: Price Bias Correction via Randomized Ads

“We algorithmically set prices based on user behavior. Can we still measure causal impact of price?”

Use randomized ad placement or feature inclusion as **instrumental variables** to estimate **true demand response** under pricing rules.

## ✅ Control Variables (W): What to Include

### 🧩 1. ****User Profile****

* prime\_status, account\_age, user\_segment

### 🧩 2. ****Shopping Context****

* category\_interest, device\_type, previous\_ads\_seen

### 🧩 3. ****Time & Geography****

* day\_of\_week, hour\_of\_day, region

### ✅ Summary Table

| **Control Variable** | **Why Include?** |
| --- | --- |
| prime\_status, region | Influence both pricing and demand |
| device\_type | May affect price delivery and purchase behavior |
| account\_age | Proxy for loyalty that affects price sensitivity |

## 🚫 What Not to Include

Avoid:

* Post-treatment variables like clicked\_after\_price\_seen

## 🧪 Code Example for W

W = df[[

'prime\_status', 'account\_age', 'user\_segment',

'device\_type', 'region', 'day\_of\_week'

]].values

Awesome! Here's the full structured explanation for:

## 🔍 Method: DynamicDML

Great question! **DynamicDML** is used when you’re studying the **time-based or sequential effects** of treatments — for example, **price changes across days**, or **staggered promotions**.

This method leverages **panel data** (multiple observations per unit over time) to model **dynamic treatment effects**, adjusting for both **unit-level confounders** and **time trends** using Double Machine Learning.

## ✅ When to Use DynamicDML at Amazon

Suppose Amazon runs a **7-day promotional campaign** for Fire TV Sticks:

“How does the promotion affect demand over time — on day 1 vs. day 7 — and does the effect persist or decay?”

Use **DynamicDML** to:

* Measure **time-dependent effects** of the same treatment,
* Separate short-term spikes from long-term impacts,
* Account for repeated observations per user/product.

## 🧠 Key Assumptions

* **Sequential Ignorability**: Treatment at time tt is independent of future potential outcomes, conditional on past covariates and outcomes.
* **No spillovers** between units (e.g., one user's treatment doesn’t affect others).
* **Panel structure**: Each unit (e.g., customer) is observed at multiple times.

## 🔧 Model Steps

1. Format your data into a **panel**: (unit, time)
2. Use ML to predict YtY\_t and TtT\_t residuals
3. Estimate dynamic treatment effect at each time tt

## ⚙️ Example Implementation

from econml.panel.dml import DynamicDML

from sklearn.ensemble import GradientBoostingRegressor

est = DynamicDML(model\_y=GradientBoostingRegressor(),

model\_t=GradientBoostingRegressor())

est.fit(Y, T, X=X, W=W, groups=customer\_ids)

theta\_t = est.const\_marginal\_effect(X\_test)

## 📈 Interpretation

* If theta\_t[3] = -1.2, then on **day 3**, a $1 discount is expected to increase purchases by **1.2 units**.

## 🛍 Amazon Case Study: Weeklong Deal on Fire Tablets

“How does the impact of a discount evolve over the course of Prime Week?”

DynamicDML reveals:

* Strongest effects on day 1 (awareness),
* Gradual decay or persistence of effects,
* Behavior of different user segments over time.

## ✅ Control Variables (W): What to Include

### 🧩 1. ****Time-Varying User Behavior****

* pages\_viewed, cart\_additions, search\_volume\_t

### 🧩 2. ****Time Dummies & Calendar Events****

* day\_of\_week, hour, prime\_day\_flag

### 🧩 3. ****Lagged Variables****

* lagged\_Y, lagged\_T, cumulative\_exposure

### ✅ Summary Table

| **Control Variable** | **Why Include?** |
| --- | --- |
| lagged\_Y | Adjust for past behavior |
| day\_of\_week | Captures seasonal variation |
| cart\_additions | Indicates short-term purchase intent |

## 🚫 What Not to Include

Avoid:

* Future outcomes (e.g., future\_Y)
* Post-treatment behaviors (e.g., reacted\_today\_after\_discount\_seen)

## 🧪 Code Example for W

W = df[[

'day\_of\_week', 'pages\_viewed', 'cart\_additions',

'lagged\_Y', 'prime\_day\_flag'

]].values

Great! Here's the full structured explanation for:

## 🔍 Method: Meta-Learners (T-, S-, X-, and DomainAdaptation Learners)

Excellent question! **Meta-learners** are flexible, modular estimators for **heterogeneous treatment effects (CATE)** using standard machine learning. They’re called “meta” because they **restructure your causal problem** into a supervised learning task.

Each learner has its strengths:

* **T-Learner**: Separate models for treated vs. control.
* **S-Learner**: Single model with treatment as a feature.
* **X-Learner**: Combines T-Learner + reweighting (great for imbalance).
* **DomainAdaptation Learner**: Adjusts for covariate shift between treated and control groups.

## ✅ When to Use Meta-Learners at Amazon

Suppose Amazon wants to **test a personalized discount** strategy on Kindle eBooks:

“How do different customer types respond to a 20% discount — and how should we tailor offers?”

Use **Meta-Learners** when:

* You want quick CATE estimates,
* You have **strong ML models** already,
* Treatment is binary (exposed vs not).

## 🧠 Key Assumptions

* **No unobserved confounding**: T⊥ε∣XT \perp \varepsilon \mid X
* **Stable Unit Treatment Value Assumption (SUTVA)**: No interference between units.
* **Overlap**: All units have chance to be treated or not.

## 🔧 Model Steps

1. Choose meta-learner type (T, S, X, etc.)
2. Train base ML models (e.g., gradient boosting)
3. Estimate CATE via transformation logic:
   * T-Learner: τ^(X)=μ^1(X)−μ^0(X)\hat{\tau}(X) = \hat{\mu}\_1(X) - \hat{\mu}\_0(X)
   * X-Learner: Adjust using propensity scores and treatment group predictions

## ⚙️ Example Implementation

from econml.metalearners import XLearner

from sklearn.ensemble import RandomForestRegressor

est = XLearner(models=RandomForestRegressor())

est.fit(Y, T, X=X)

tau = est.effect(X\_test)

## 📈 Interpretation

* If tau = 2.5 for a customer segment, applying a discount is estimated to **increase their spending by $2.50**.

## 🛍 Amazon Case Study: Personalized Kindle Discounts

“Which customers are most likely to respond to a 20% discount on eBooks?”

Use X-Learner to:

* Capture heterogeneity in effect,
* Compare Prime vs non-Prime response,
* Prioritize segments for future campaigns.

## ✅ Control Variables (X): What to Include

(Note: Meta-learners typically don’t use W and instead rely entirely on **X**)

### 🧩 1. ****User Characteristics****

* account\_age, prime\_status, user\_segment

### 🧩 2. ****Purchase Behavior****

* past\_spend\_ebooks, pages\_viewed, device\_type

### 🧩 3. ****Context Features****

* day\_of\_week, campaign\_id, discount\_level

### ✅ Summary Table

| **Feature** | **Why Include?** |
| --- | --- |
| account\_age, segment | Drives how users respond to price changes |
| device\_type | Kindle users may be more discount-sensitive |
| campaign\_id | Controls for experiment context |

## 🚫 What Not to Include

Avoid:

* Post-treatment metrics: clicked\_discount\_banner, time\_spent\_after\_offer

## 🧪 Code Example for X

X = df[[

'account\_age', 'prime\_status', 'user\_segment',

'device\_type', 'past\_spend\_ebooks', 'campaign\_id'

]].values

Awesome! Here's the full structured explanation for:

## 🔍 Method: OrthoForest

Great question! **OrthoForest** is a powerful method for estimating **heterogeneous treatment effects** in **high-dimensional, nonlinear settings**. It’s built on the principles of **orthogonalization** (from Double Machine Learning) and **local forest models** to deliver **nonparametric CATEs** with robustness to overfitting and nuisance estimation error.

It’s especially useful when:

* You expect **complex interactions** between features and treatment effects,
* You want **reliable effect estimates across many user segments**.

## ✅ When to Use OrthoForest at Amazon

Suppose Amazon is testing **a dynamic discount system** for Alexa devices:

“How does a $10 discount affect different customer types — especially when we have hundreds of behavioral features?”

Use OrthoForest to:

* Estimate personalized price response,
* Handle many covariates (product views, site activity, user profile),
* Produce **locally smoothed** CATEs with variance bounds.

## 🧠 Key Assumptions

* **Unconfoundedness**: T⊥ε∣X,WT \perp \varepsilon \mid X, W
* **Smoothness**: τ(X)\tau(X) changes gradually in the covariate space.
* **Sufficient overlap** across treatment and control in feature space.

## 🔧 Model Steps

1. Fit ML models for Y∼X,WY \sim X, W and T∼X,WT \sim X, W to residualize.
2. Fit a **locally adaptive causal forest** on the residuals.
3. Use influence function scores to stabilize local treatment estimates.

## ⚙️ Example Implementation

from econml.orf import DMLOrthoForest

from sklearn.ensemble import RandomForestRegressor

est = DMLOrthoForest(model\_t=RandomForestRegressor(),

model\_y=RandomForestRegressor(),

n\_trees=1000,

min\_leaf\_size=10)

est.fit(Y, T, X=X, W=W)

tau = est.effect(X\_test)

intervals = est.effect\_interval(X\_test)

## 📈 Interpretation

* If tau = -1.8, this user’s estimated demand increases by 1.8 units per $1 price drop — based on similar users nearby in the feature space.

## 🛍 Amazon Case Study: Alexa Device Demand by Behavioral Profile

“Can we identify which user profiles are most price sensitive — even when we don’t know the right variables in advance?”

OrthoForest can:

* Handle hundreds of covariates (e.g., product scroll depth, session history),
* Estimate **nonlinear treatment-response functions**,
* Guide pricing tiers or dynamic offer targeting.

## ✅ Control Variables (W): What to Include

### 🧩 1. ****User Activity Features****

* search\_volume, session\_length, click\_depth, product\_views

### 🧩 2. ****Profile & Device****

* prime\_status, signup\_date, device\_type, browser\_type

### 🧩 3. ****Offer Context****

* campaign\_id, offer\_seen, inventory\_day

### ✅ Summary Table

| **Control Variable** | **Why Include?** |
| --- | --- |
| click\_depth, session\_length | User engagement drives both price and outcome |
| browser\_type, device\_type | Interface affects exposure and conversion |
| offer\_seen | Controls for visibility of the price |

## 🚫 What Not to Include

Avoid:

* Mediators (e.g., time\_spent\_after\_offer\_shown)
* Variables that encode post-treatment behavior

## 🧪 Code Example for W

W = df[[

'search\_volume', 'session\_length', 'click\_depth',

'prime\_status', 'signup\_date', 'offer\_seen'

]].values

Excellent! Here's the full structured explanation for:

## 🔍 Method: DeepIV

Great question! **DeepIV** combines **instrumental variables** with **deep learning** to estimate **causal effects** in the presence of endogeneity and **complex nonlinear relationships**.

It’s particularly helpful when:

* Your treatment (e.g., price) is **endogenous**,
* You suspect nonlinear interactions between treatment and outcome,
* You have **rich, high-dimensional features**, and
* You can identify an instrument that affects the treatment but not the outcome directly.

## ✅ When to Use DeepIV at Amazon

Let’s say Amazon wants to understand how **dynamic pricing affects conversion** on smart home products:

“But price is optimized algorithmically based on user behavior — so it’s endogenous.”

If Amazon runs a **randomized ad or deal exposure** that affects price but **not demand directly**, that can serve as an instrument.

Use DeepIV to:

* Correct for endogeneity using the instrument,
* Capture nonlinear demand curves,
* Scale with large behavioral datasets.

## 🧠 Key Assumptions

* **Instrument Relevance**: Cov(Z,T)≠0\text{Cov}(Z, T) \neq 0
* **Exclusion Restriction**: Z⊥ε∣XZ \perp \varepsilon \mid X; the instrument affects YY only via TT
* **Monotonicity** (weaker, sometimes assumed): Direction of effect is consistent

## 🔧 Model Steps

1. Train a **deep neural network** to model T∼Z,XT \sim Z, X — the treatment assignment mechanism.
2. Sample treatment values from this network to simulate potential exposures.
3. Train another **neural network** to model Y∼T,XY \sim T, X, using simulated treatments.

## ⚙️ Example Implementation

from econml.iv.nnet import DeepIV

est = DeepIV(n\_components=10, # latent treatment samples

m=2, # treatment dimension

n\_epochs=20,

verbose=1)

est.fit(Y, T, Z=Z, X=X)

theta = est.effect(X\_test)

## 📈 Interpretation

* If theta = -2.0, reducing price by $1 is expected to increase purchases by 2 units, **even when prices are not randomized**.

## 🛍 Amazon Case Study: Algorithmic Pricing on Echo Buds

“We want to estimate the true demand response to dynamic price offers — but prices are algorithmically personalized.”

Solution:

* Use **randomized banner or promo view** as instrument,
* Apply DeepIV to estimate **nonlinear elasticity curves**,
* Correct for algorithmic bias.

## ✅ Control Variables (X and W): What to Include

### 🧩 1. ****User Context****

* prime\_status, region, device\_type

### 🧩 2. ****Historical Behavior****

* avg\_cart\_value, prior\_ad\_clicks, product\_page\_visits

### 🧩 3. ****Campaign Metadata****

* campaign\_type, ad\_placement, week\_of\_year

### ✅ Summary Table

| **Control Variable** | **Why Include?** |
| --- | --- |
| product\_page\_visits | Correlates with both price exposure and intent |
| campaign\_type, region | Control for contextual variation |
| prior\_ad\_clicks | Affects targeting algorithm |

## 🚫 What Not to Include

Avoid:

* Post-treatment behavior (e.g., checkout\_conversion\_after\_price)
* Direct mediators from price to outcome

## 🧪 Code Example for W

W = df[[

'prime\_status', 'region', 'device\_type',

'avg\_cart\_value', 'prior\_ad\_clicks', 'campaign\_type'

]].values

Perfect! Here's the structured explanation for:

## 🔍 Method: SieveTSLS (Sieve Two-Stage Least Squares)

Great question! **SieveTSLS** is a nonparametric extension of the classical instrumental variable (IV) method. It’s designed to estimate **nonlinear treatment effects** using **basis function expansions** (e.g., splines, polynomials), while still leveraging valid IVs to handle **endogeneity**.

This method is especially helpful when:

* You expect a **smooth, nonlinear relationship** between treatment and outcome (like a price-demand curve),
* You have a **valid instrument** for treatment,
* You want **transparency and interpretability** in your causal model.

## ✅ When to Use SieveTSLS at Amazon

Let’s say Amazon wants to explore:

“Is there a **threshold effect** where demand sharply drops if price exceeds $29.99 for Echo Buds?”

You suspect price-demand relationship isn’t linear, and price is set algorithmically (hence endogenous). You also have a **randomized ad feature** as an instrument.

SieveTSLS allows:

* Correction for endogeneity,
* Flexible modeling of smooth nonlinear effects,
* **Interpretable response curves**.

## 🧠 Key Assumptions

* **Instrument Validity**: Z⊥ε∣XZ \perp \varepsilon \mid X; Z→T→YZ \rightarrow T \rightarrow Y
* **Smoothness**: Treatment effect function is continuous and can be well-approximated by basis functions.
* **Overlap**: Treatment and instrument vary enough in the data.

## 🔧 Model Steps

1. **Stage 1**: Regress treatment T∼Z,XT \sim Z, X using basis expansion
2. **Stage 2**: Regress outcome Y∼T,XY \sim T, X using the predicted TT

## ⚙️ Example Implementation

from econml.iv.sieve import SieveTSLS

from sklearn.preprocessing import PolynomialFeatures

est = SieveTSLS(degree=3) # cubic polynomial basis

est.fit(Y, T, Z=Z, X=X)

theta\_curve = est.const\_marginal\_effect(X\_test)

## 📈 Interpretation

* You get a **smooth curve** of treatment effects over price levels.
* If the slope of the curve drops steeply at $29.99, it suggests a **psychological price threshold**.

## 🛍 Amazon Case Study: Nonlinear Price Threshold on Echo Buds

“How does demand change across price points? Is there a drop at $30?”

Use SieveTSLS to:

* Model smooth demand response,
* Adjust for bias from strategic pricing,
* Guide optimal price setting based on elasticities.

## ✅ Control Variables (X): What to Include

### 🧩 1. ****User Context****

* prime\_status, region, account\_age

### 🧩 2. ****Platform Engagement****

* device\_type, click\_depth, visit\_frequency

### 🧩 3. ****Price Visibility & Position****

* page\_position, ad\_exposure, stock\_level

### ✅ Summary Table

| **Control Variable** | **Why Include?** |
| --- | --- |
| account\_age, visit\_frequency | Adjust for baseline shopping behavior |
| page\_position, ad\_exposure | Affects how users perceive prices |
| region, device\_type | Segment-specific pricing patterns |

## 🚫 What Not to Include

Avoid:

* Outcomes after pricing: abandon\_rate\_after\_offer
* Variables affected by price directly

## 🧪 Code Example for X

X = df[[

'prime\_status', 'region', 'device\_type',

'click\_depth', 'visit\_frequency', 'page\_position'

]].values

Amazing! Let’s wrap up with the final method:

## 🔍 Method: FederatedEstimator

Great question! **FederatedEstimator** is used when your data is **distributed across multiple sources** — such as **regions, business units, or countries** — and **you can’t pool the raw data** due to **privacy, regulatory, or infrastructure constraints**.

It allows each site (or "client") to **train its own local model**, and then aggregates them into a **single federated model** — without sharing the raw data.

## ✅ When to Use FederatedEstimator at Amazon

Let’s say you want to measure the effect of a **$10 discount on Fire TV Sticks**:

“But data is stored separately for Amazon’s US, EU, and Asia sites, and **can’t be shared due to GDPR**.”

FederatedEstimator enables:

* Local training of estimators (e.g., LinearDML),
* Central aggregation of causal effects,
* **No user-level data sharing**.

## 🧠 Key Assumptions

* **Same estimator structure** across all nodes.
* **Sufficient sample size** at each client.
* **No leakage or duplication of observations** across datasets.

## 🔧 Model Steps

1. Define a **base estimator** (e.g., LinearDML).
2. Fit the estimator independently at each site: US, EU, Asia.
3. Aggregate the trained models using FederatedEstimator.

## ⚙️ Example Implementation

from econml.federated\_learning import FederatedEstimator

from econml.dml import LinearDML

# Fit each region locally

us\_est = LinearDML(...).fit(Y\_us, T\_us, X=X\_us, W=W\_us)

eu\_est = LinearDML(...).fit(Y\_eu, T\_eu, X=X\_eu, W=W\_eu)

asia\_est = LinearDML(...).fit(Y\_asia, T\_asia, X=X\_asia, W=W\_asia)

# Combine

fed\_est = FederatedEstimator([us\_est, eu\_est, asia\_est])

theta\_global = fed\_est.effect(X\_test)

## 📈 Interpretation

* theta\_global = -1.0 means the average effect across all regions is a 1-unit increase in demand for every $1 discount.
* You can still access **region-specific** estimates using the original estimators.

## 🛍 Amazon Case Study: Global Impact of Fire TV Discount

“Can we estimate a global treatment effect without moving user-level data between data centers?”

FederatedEstimator:

* Trains locally at each regional office,
* Aggregates insights for corporate marketing teams,
* Complies with **GDPR and internal data governance**.

## ✅ Control Variables (W): What to Include

### 🧩 1. ****User Demographics & Region****

* region, language, account\_age

### 🧩 2. ****Device & Platform****

* device\_type, login\_status, app\_version

### 🧩 3. ****Marketing & Product Context****

* promo\_seen, stock\_level, shipping\_speed

### ✅ Summary Table

| **Control Variable** | **Why Include?** |
| --- | --- |
| region, language | Regional variation in response |
| device\_type | Affects how promotions are delivered |
| promo\_seen, stock | Availability and awareness adjust expectations |

## 🚫 What Not to Include

Avoid:

* Any features that would require centralizing sensitive user data
* Post-treatment behaviors (e.g., time\_on\_page\_after\_offer)

## 🧪 Code Example for W

W = df[[

'region', 'language', 'account\_age',

'device\_type', 'promo\_seen', 'stock\_level'

]].values