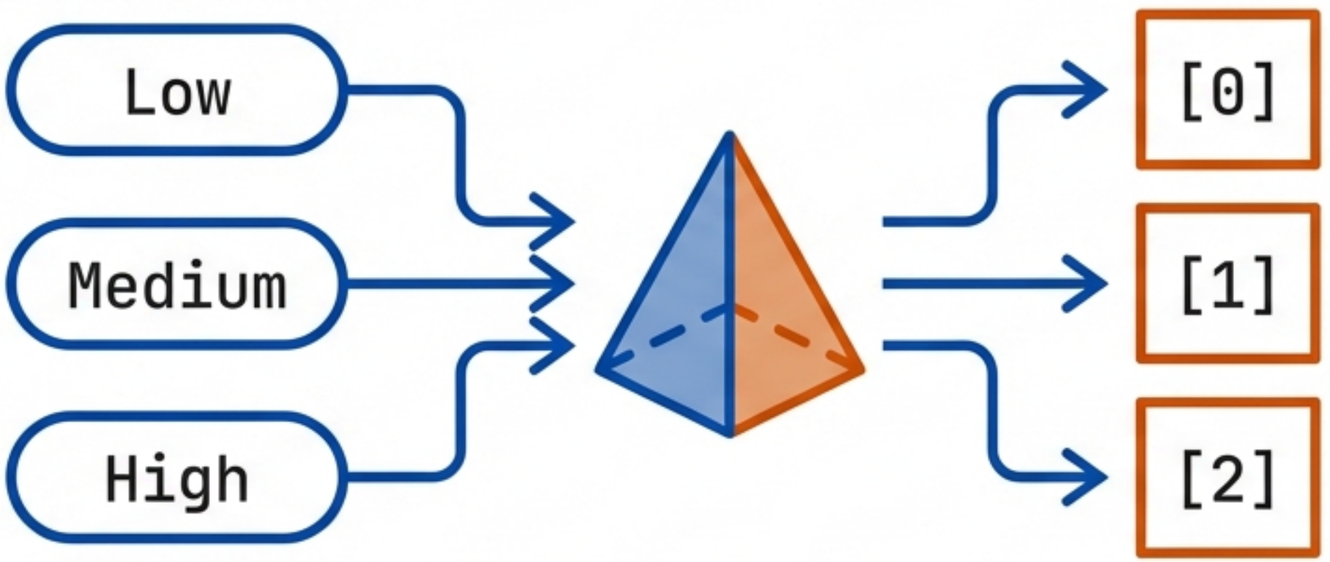
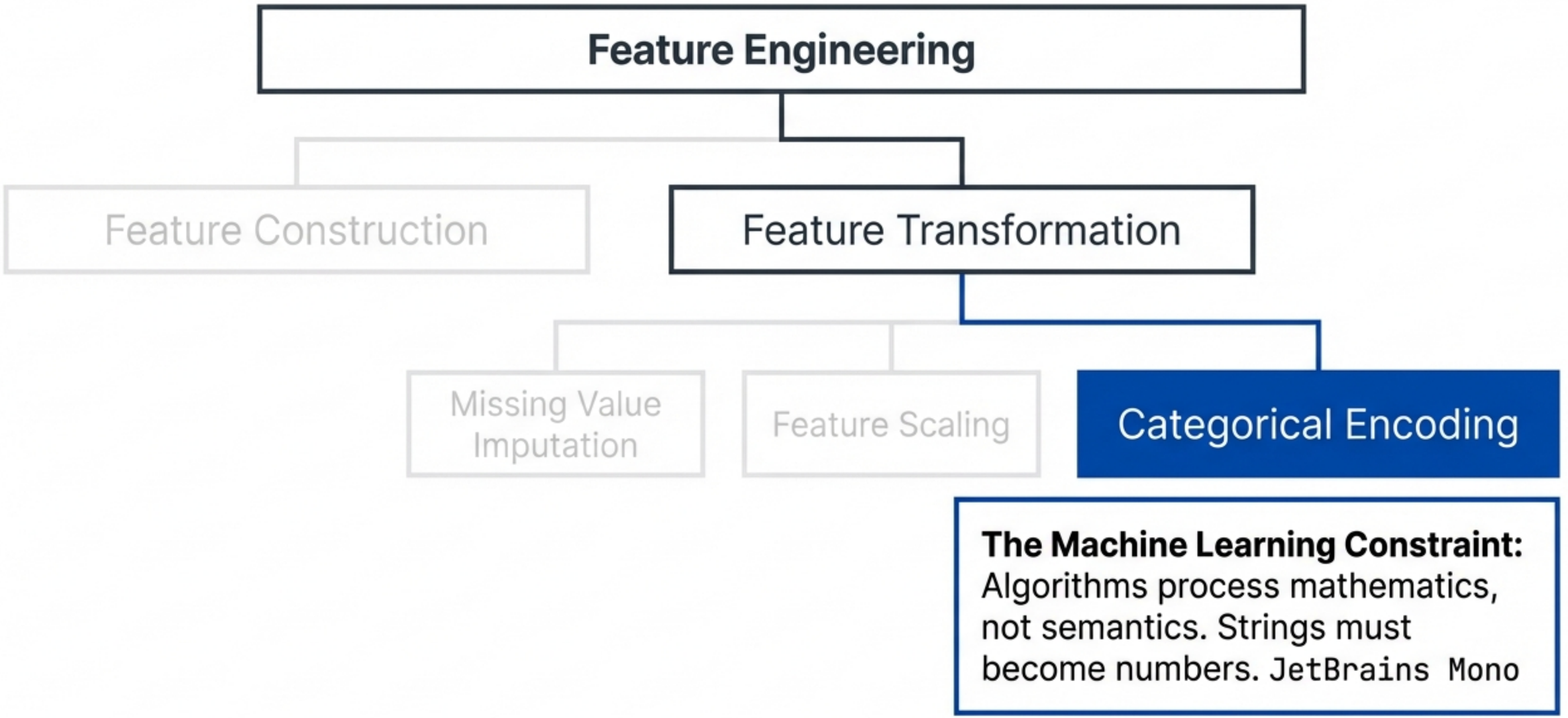
Encoding Categorical Data

Ordinal & Label Encoding



A strategic guide to transforming qualitative features into machine-readable formats using Scikit-Learn.
Part of the Feature Engineering & Transformation Series.

Mapping the Feature Engineering Landscape



Diagnosing Data: Nominal vs. Ordinal Attributes

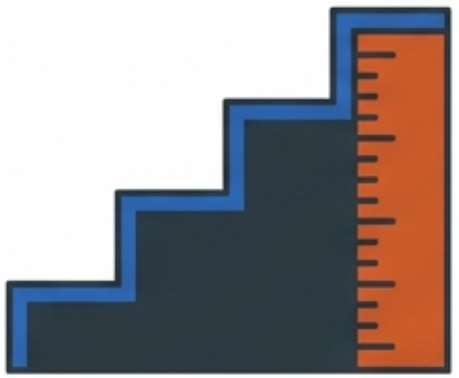
Nominal Data

Definition: Categories with no intrinsic order. Mathematical inequality does not apply.

- States (`West Bengal, Maharashtra`)
- Gender (`Male, Female`)
- Branch (`CSE, IT`)

`Maharashtra > West Bengal = FALSE`

Ordinal Data



Definition: Categories possessing a definitive, inherent rank or hierarchy.

- Education (`High School < UG < PG`)
- Reviews (`Poor < Average < Good`)

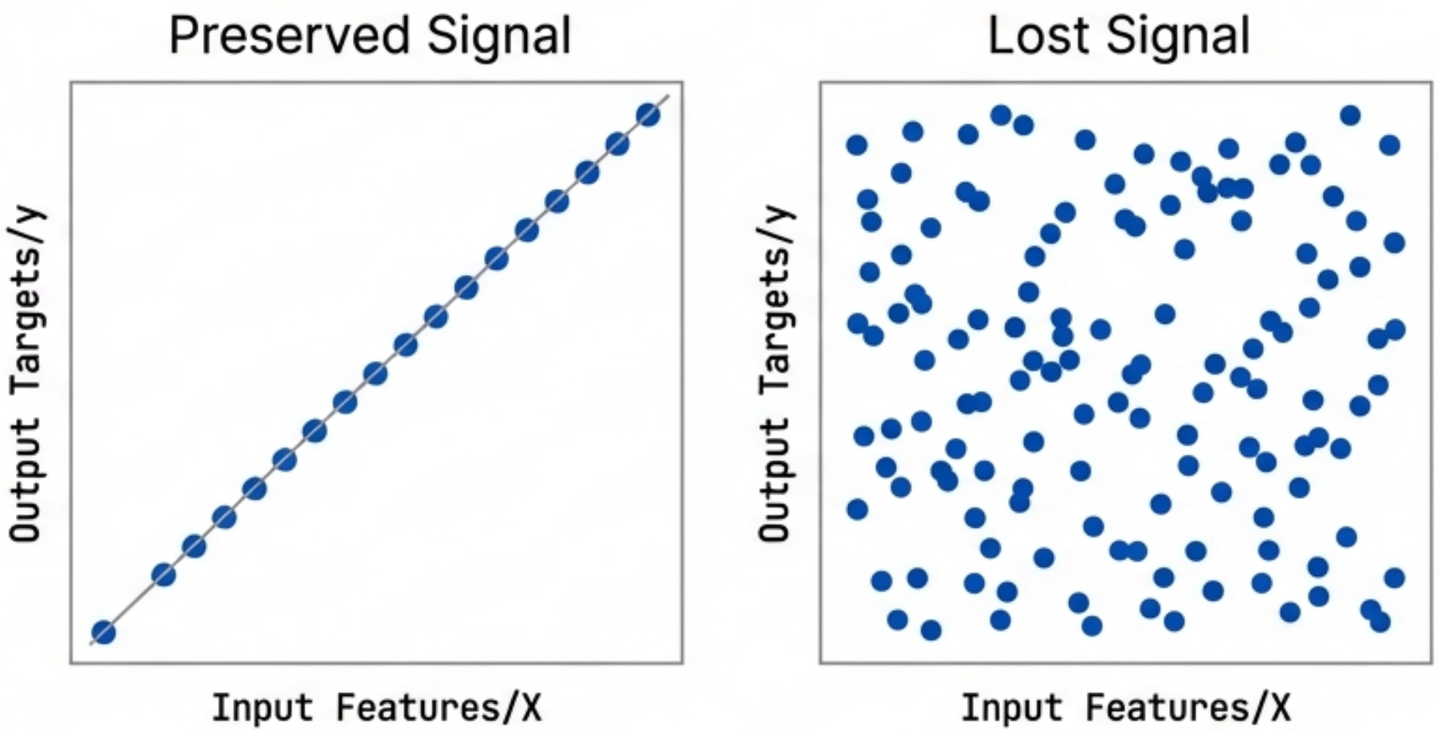
`PG > UG > High School = TRUE`

The Ordinal Imperative

Excellent (2) > Good (1) > Poor (0)

When data has rank, the encoding must reflect that rank mathematically. Randomly assigning numbers (e.g., Poor=5, Good=1) destroys the feature's information signal.

We must enforce the hierarchy manually to preserve the "intelligence" of the variable.

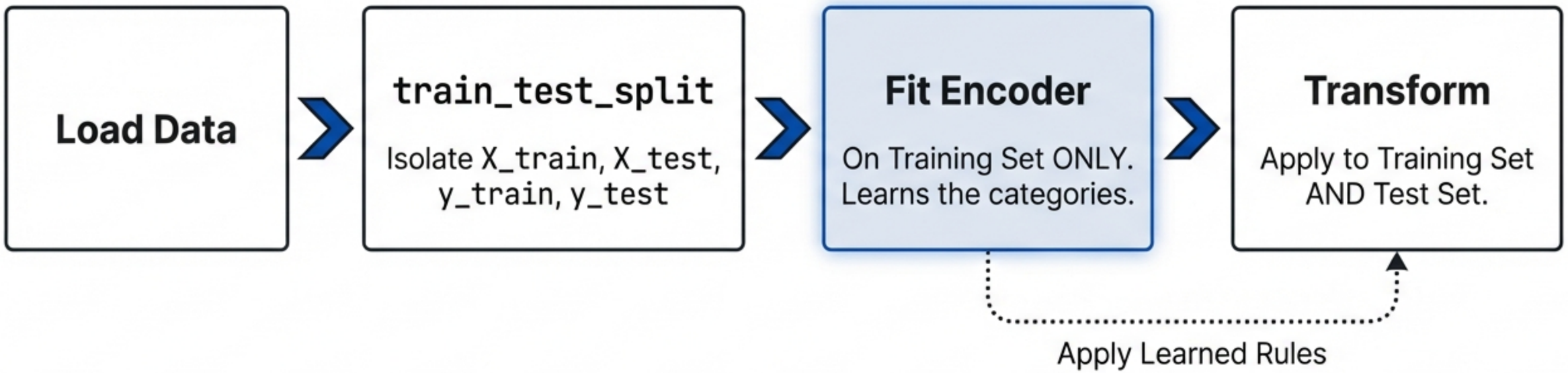


The Scikit-Learn Decision Matrix

`OrdinalEncoder`	`LabelEncoder`
Use Case: `Input Features (X)`	**Use Case:** `Output Targets (y) ONLY`
Data Type: `Ordinal Categorical` (e.g., `Education, Quality`)	**Data Type:** `Categorical Targets` (e.g., `'Yes/No', 'Setosa/Versicolor'`)
Function: Transforms features while allowing **user-defined ordering** to preserve rank.	**Function:** Encodes **target classes** between `0` and `n-1`.
Syntax: `sklearn.preprocessing.OrdinalEncoder`	**Syntax:** `sklearn.preprocessing.LabelEncoder`

WARNING: Scikit-Learn Documentation Explicitly States: `LabelEncoder should not be used for input features (X).`

The Workflow: Split, Then Transform



This prevents data leakage by ensuring the model is evaluated on data structures it hasn't "seen" during the fitting process.

Case Study: Customer Purchasing Data

	Age	Gender	Review	Education	Purchased
1	30	Female	Average	School	No
2	68	Female	Poor	UG	No
3	45	Male	Good	PG	Yes

Numerical (Ignore)

Nominal (Male/Female). Out of Scope.

Ordinal Data. Action: Use OrdinalEncoder.

Target/Label. Action: Use LabelEncoder.

Implementing OrdinalEncoder

```
from sklearn.preprocessing import OrdinalEncoder

# Define order: Poor < Average < Good; School < UG < PG
oe = OrdinalEncoder(categories=[
    ['Poor', 'Average', 'Good'],
    ['School', 'UG', 'PG']
])

oe.fit(X_train)
X_train_transformed = oe.transform(X_train)
```

The 'categories' Parameter. We manually pass a **list of lists to strictly enforce the order.** If omitted, Scikit-Learn assigns numbers randomly or alphabetically (e.g., Poor might become 2 instead of 0), destroying the hierarchy.

Ordinal Transformation Results

Raw Data (String)

Review	Education
Good	UG
Poor	PG

Transformed (Float)

Review	Education
2.0	1.0
0.0	2.0

The string data has been successfully converted to floats that preserve the magnitude of the original categories (Good > Poor).

Implementing LabelEncoder

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
le.fit(y_train)

y_train_transformed = le.transform(y_train)
y_test_transformed = le.transform(y_test)
```

- **Target Only:** Applied to the `'Purchased'` column.
- **Auto-Assignment:** No parameters passed. The encoder **assigns integers `0` to `n-1` automatically.**
- **Result:** `'No'` becomes `0`, `'Yes'` becomes `1`.

The Limitation: Nominal Data

False Relationship.

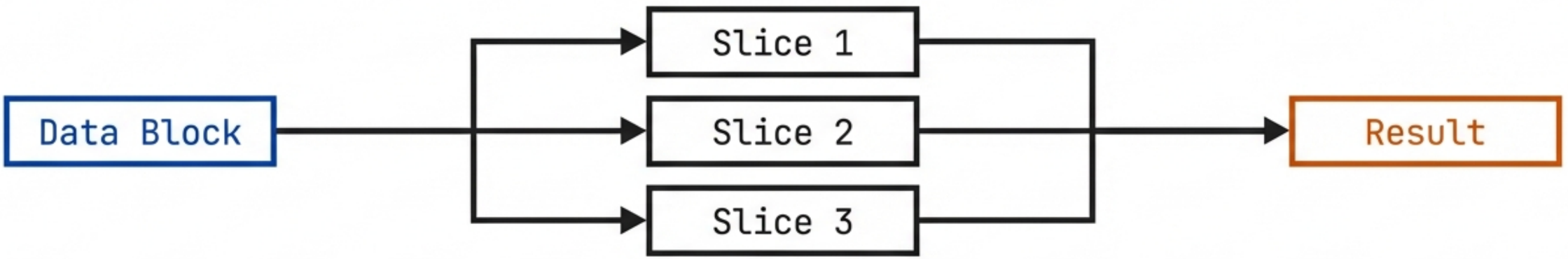
Using Ordinal Encoding on nominal data (States, Gender, Colors) introduces false mathematical relationships. The model will infer rank where none exists.

Solution: Nominal data requires **One-Hot Encoding.**

This creates binary dummy variables and will be covered in the next module.

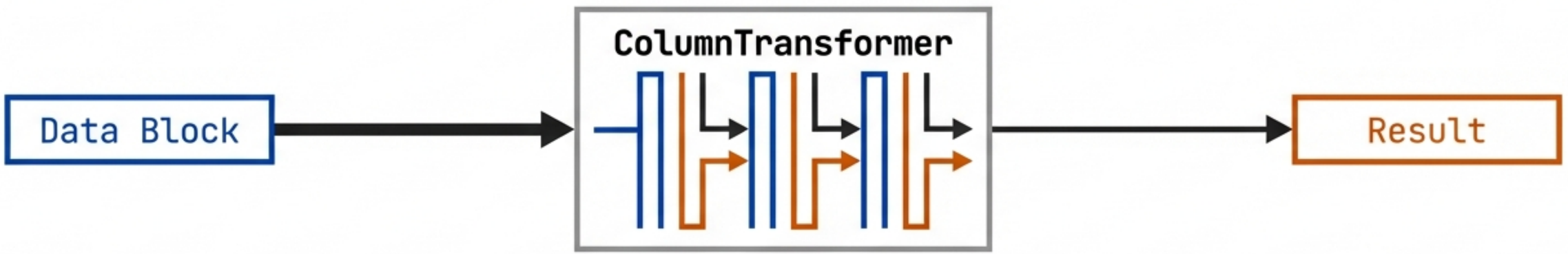
Scaling the Workflow: The Column Transformer

Current / Manual



Tedious & Error Prone

Professional / Automated



Streamlined

In production pipelines, we avoid manual slicing. The '**ColumnTransformer**' class applies different transforms to different columns simultaneously in a single step.

Encoding Cheatsheet

OrdinalEncoder

Target: Input Features (X).
Data: Ordered Categories.
Key Syntax: `categories=[['Low', 'High']]`.

LabelEncoder

Target: Output Labels (y).
Data: Classification Classes.
Key Syntax: `LabelEncoder()` (No args).

Nominal Data

Target: Input Features (X).
Data: Unordered (States, Colors).
Tool: One-Hot Encoding (Do NOT use Ordinal).

Workflow

1. Split Data
2. Fit (Train only)
3. Transform (Train & Test).