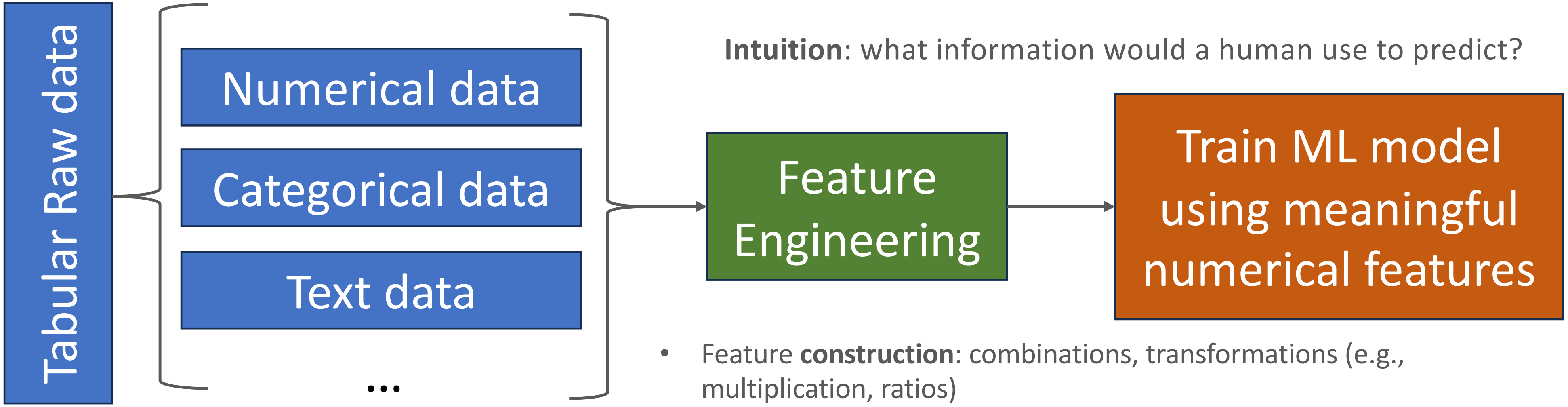
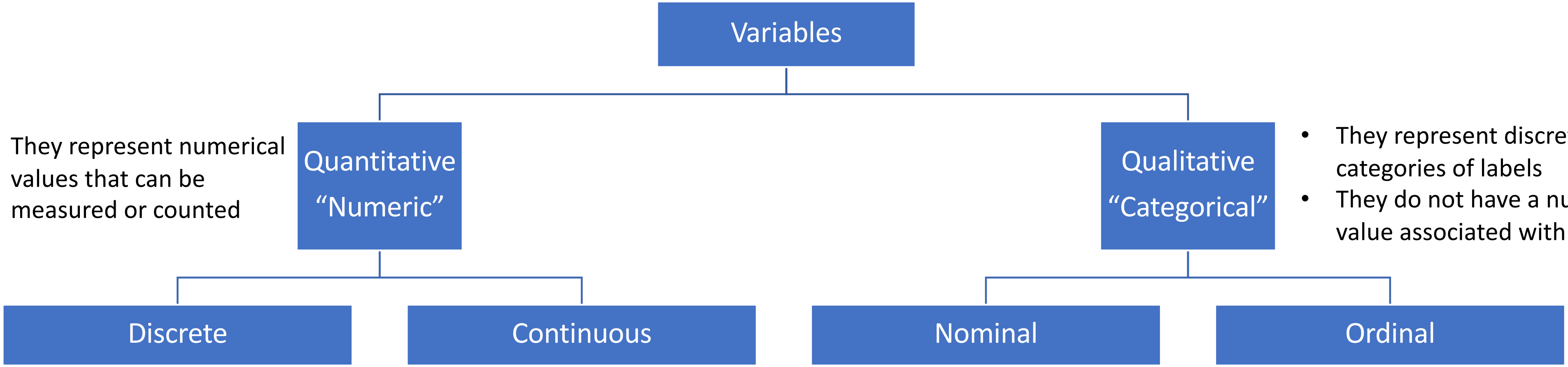
Feature engineering

Use **domain knowledge** and **data understanding** to create meaningful features from raw data for ML models



Intuition: what information would a human use to predict?

- Feature **construction**: combinations, transformations (e.g., multiplication, ratios)
- Feature **extraction**: encoding, text vectorization
- Feature **selection**: correlation filtering, dimensionality reduction (PCA)



Variables

Quantitative “Numeric”

They represent numerical values that can be measured or counted

Qualitative “Categorical”

- They represent discrete categories of labels
- They do not have a numerical value associated with them

Discrete

They are whole numbers (not a fractional number) than can be positive, negative, or zero

- **Number of:**
  - **Children in a Family:** 0, 1, 2, …
  - **Students in a classroom:** 10, 20, …
  - **Cars in a parking lot**: 0, 1, 2, …

Continuous

They are values within a given range., and they often have an infinite number of possible values

- **Height:** 180.5cm, …
- **Weight**: 54.6kg, …
- **Temperature**: 30.2°C, …
- **Time**: 5.3 seconds, …
- **Age**: 22.4 years, …
  - We often represent age in whole numbers (e.g., 22 years)

Nominal

They do not have a natural order

- **Colors:** red, blue, …
- **States:** Pichincha, Azuay, ….
- **Zip codes:** 17001, 17005, ...
- **Blood groups:** A, B, AB, O+, ...

Ordinal

There is an order or ranking but the intervals between the categories are not necessary uniform

- **Education level**: bachelor’s, master’s
- **Education grades**: A, B, C
- **Socioeconomic status**: lower, middle, upper
- **Customer satisfaction**: very dissatisfied, dissatisfied, neutral, satisfied, very satisfied
- **Likert scales**: strongly disagree, disagree, neutral, agree, strongly agree

Categorical data

Categorical features:

- They do not have a natural numerical representation
- Most ML models require converting categorical features to numerical ones.
- Example:
  - $color \in \{green, red, blue\}$,
  - $isFraud \in \{false, true\}$

Encoding categorical features

Encode/define a mapping:

Assign a number to each category

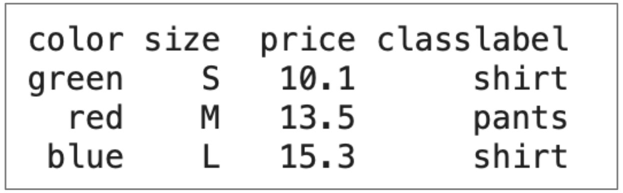
- Ordinals: categories are ordered. E.g.,: $\text{size} \in \{S < M < L\}$, we can assign $\mathrm{S} = 1, \mathrm{M} = 2, \mathrm{L} = 3$
- Nominals: categories are unordered. E.g.,: $color \in \{green, red, blue\}$, we can assign numbers randomly

Encoding categorical features

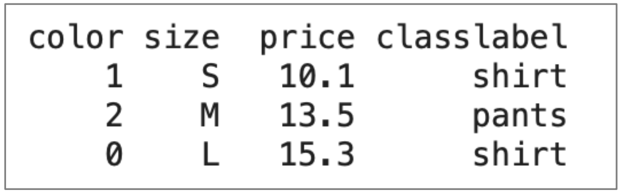
LabelEncoder: It encodes *target labels* (y) or one feature only (not the input X) with value between *0 and (n_classes-1)*

- Can be used to transform non-numerical labels or numerical labels

```
LabelEncoder().fit_transform(df['color'])
```



```
color size  price classlabel
green    S   10.1      shirt
  red    M   13.5      pants
 blue    L   15.3      shirt
```



```
color size  price classlabel
    1    S   10.1      shirt
    2    M   13.5      pants
    0    L   15.3      shirt
```

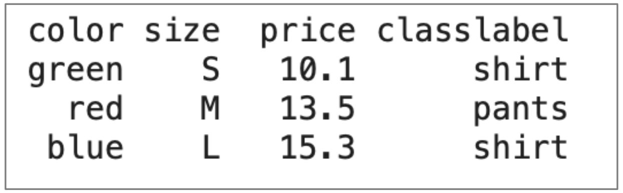
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

Encoding categorical features

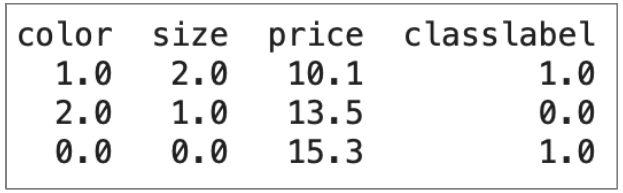
OrdinalEncoder: It encodes categorical features as an integer array

- Encodes (two or more) categorical features (It does not work on one feature)
- Returns a single column of integers between *0 to (n_categories-1)* per feature

```
OrdinalEncoder().fit_transform(df[['color','size','classlabel']])
```



```
color size  price classlabel
green    S   10.1      shirt
  red    M   13.5      pants
 blue    L   15.3      shirt
```



```
color  size  price  classlabel
  1.0   2.0   10.1         1.0
  2.0   1.0   13.5         0.0
  0.0   0.0   15.3         1.0
```

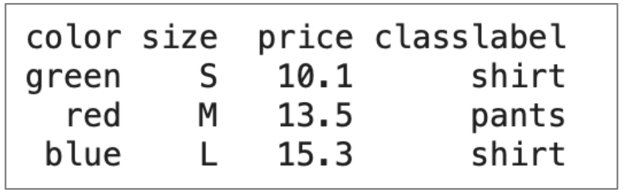
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html#sklearn.preprocessing.OrdinalEncoder

Encoding categorical features

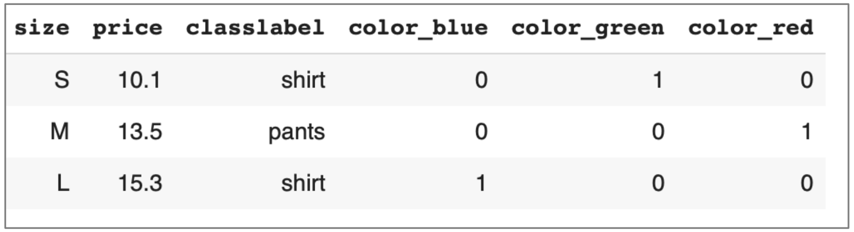
OneHotEncoder: It encodes categorical features as a one-hot numerical array

- Explode the categorical features into many binary features (*as many categories per feature*)
- Works on two or more features

```
OneHotEncoder(sparse_output=False, handle_unknown='ignore').fit_transform(df_3[['color']])
```



```
color size  price classlabel
green    S   10.1      shirt
  red    M   13.5      pants
 blue    L   15.3      shirt
```



size	price	classlabel	color_blue	color_green	color_red
S	10.1	shirt	0	1	0
M	13.5	pants	0	0	1
L	15.3	shirt	1	0	0

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html

	Label Encoding	Ordinal Encoding	One-Hot Encoding
Description	Assigns a unique integer to each category. No assumptions about order.	Assigns a unique number to each category.	Converts each category to a new column with binary values (0 or 1).
Common Use	*When working with categorical data without order*	*When categories have an implicit order.*	*When categories have **no** inherent order.*
Risk	Can mislead models that assume ordinal relationships due to numeric values	May be misleading if the model assumes numbers have an order (0, 1, 2, etc.).	Avoids the category order problem
Application	Best with Tree-based models, SVM, KNN. Avoid with linear models.	Works best with decision trees, SVM, KNN (where order can make sense).	Best for algorithms that do not assume relationships between categories (Logistic Regression, Neural Networks).
Number of Columns	Does not increase number of columns — replaces with a single numeric feature	Does not increase the number of columns.	Increases the number of columns based on the number of categories.
Example	Red = 0, Green = 1, Blue = 2	Low = 1, Medium = 2, High = 3	Red = [1, 0, 0], Green = [0, 1, 0], Blue = [0, 0, 1]