

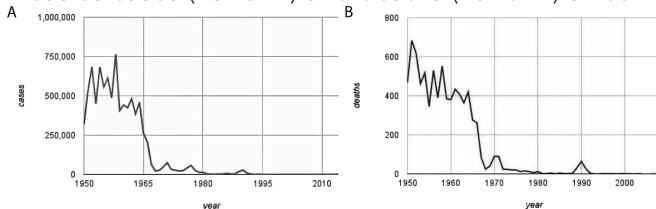
Encoding Categorical Variables as Quantitative

Dennis Sun (modified by Jonathan Taylor)
Stanford University



April 15, 2026

US measles cases (Panel A) and deaths (Panel B) since 1950



(Source: US Centers for Disease Control and Prevention)



① Motivation

② Encoding Categorical Variables

③ Column Transformations in Scikit-Learn



Review

```
features = ["Gr Liv Area", "Bedroom AbvGr", "Full Bath", "Half Bath"]  
df[features]
```

	Gr Liv Area	Bedroom AbvGr	Full Bath	Half Bath
0	1656	3	1	0
1	896	2	1	0
2	1329	3	1	1
3	2110	3	2	1
4	1629	3	2	1
...
2925	1003	3	1	0
2926	902	2	1	0
2927	970	3	1	0
2928	1389	2	1	0
2929	2000	3	2	1

2930 rows x 4 columns

We might want to scale
the variables first!

Last class, we discussed how to measure the distance between two observations \mathbf{x} and \mathbf{x}' .

For example, we can calculate the Euclidean (ℓ_2) distance:

$$d(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{j=1}^m (x_j - x'_j)^2}.$$



What if there are categorical variables?

```
features = ["Gr Liv Area", "House Style", "Bedroom AbvGr",  
            "Full Bath", "Half Bath", "Neighborhood"]  
df[features]
```

	Gr Liv Area	House Style	Bedroom AbvGr	Full Bath	Half Bath	Neighborhood
0	1656	1Story	3	1	0	NAmes
1	896	1Story	2	1	0	NAmes
2	1329	1Story	3	1	1	NAmes
3	2110	1Story	3	2	1	NAmes
4	1629	2Story	3	2	1	Gilbert
...
2925	1003	SLvl	3	1	0	Mitchel
2926	902	1Story	2	1	0	Mitchel
2927	970	SFoyer	3	1	0	Mitchel
2928	1389	1Story	2	1	0	Mitchel
2929	2000	2Story	3	2	1	Mitchel

2930 rows x 6 columns

If we want to calculate distances, we need to convert the categorical variables into quantitative variables first!



1 Motivation

2 Encoding Categorical Variables

3 Column Transformations in Scikit-Learn



Encoding Categorical Variables as Quantitative

There is a standard way to encode a categorical variable as a quantitative variable: **dummy encoding** or **one-hot encoding**.

House Style		House Style_1.5Fin	House Style_1.5Unf	House Style_1Story	House Style_2.5Fin	House Style_2.5Unf	House Style_2Story	House Style_SFoyer	House Style_SLvl
0	1Story	0	0	1	0	0	0	0	0
1	1Story	0	0	1	0	0	0	0	0
2	1Story	0	0	1	0	0	0	0	0
3	1Story	0	0	1	0	0	0	0	0
4	2Story	0	0	0	0	0	1	0	0
...
2925	SLvl	0	0	0	0	0	0	0	1
2926	1Story	0	0	1	0	0	0	0	0
2927	SFoyer	0	0	0	0	0	0	1	0
2928	1Story	0	0	1	0	0	0	0	0
2929	2Story	0	0	0	0	0	1	0	0

2930 rows x 1 columns 2930 rows x 8 columns

- Each class gets its own column.
 - Each column consists of 0s and 1s. A 1 indicates that the observation was in that class.
- 1 How many 1s are in each row?
 - 2 How many 1s are in each column?



Dummy Encoding in Pandas

Let's go into Colab to learn how to do dummy encoding in Pandas.



1 Motivation

2 Encoding Categorical Variables

3 Column Transformations in Scikit-Learn



Dummy Encoding in Scikit-Learn

We can do dummy encoding in Scikit-Learn using `OneHotEncoder`.

```
from sklearn.preprocessing import OneHotEncoder

# declare the encoder
enc = OneHotEncoder()

# fit the encoder to data
enc.fit(df[["House Style"]])

# transform the data
enc.transform(df[["House Style"]])
<2930x8 sparse matrix of type '<class 'numpy.float64''>'
  with 2930 stored elements in Compressed Sparse Row format>
```

Huh, what's a "sparse matrix"?



Dummy Encoding in Scikit-Learn

We can cast a sparse matrix to a “dense” one using `.toarray()`...
...or specify that we don't want a sparse matrix to begin with.

```
from sklearn.preprocessing import OneHotEncoder
```

```
# declare the encoder
```

```
enc = OneHotEncoder(sparse_output=False)
```

```
# fit the encoder to data
```

```
enc.fit(df[["House Style"]])
```

```
# transform the data
```

```
enc.transform(df[["House Style"]])
```

```
array([[0., 0., 1., ..., 0., 0., 0.],  
       [0., 0., 1., ..., 0., 0., 0.],  
       [0., 0., 1., ..., 0., 0., 0.],  
       ...,  
       [0., 0., 0., ..., 0., 1., 0.],  
       [0., 0., 1., ..., 0., 0., 0.],  
       [0., 0., 0., ..., 1., 0., 0.]])
```



Mixed Variables in Scikit-Learn

What if we have a mix of quantitative and categorical variables, and we only want to dummy encode the categorical ones?

We make a `ColumnTransformer`.

```
from sklearn.compose import make_column_transformer

transformer = make_column_transformer(
    (OneHotEncoder(), ["House Style", "Neighborhood"]),
    remainder="passthrough")
transformer.fit(df[features])
transformer.transform(df[features])
<2930x40 sparse matrix of type '<class 'numpy.float64''>'
    with 15717 stored elements in Compressed Sparse Row format>
```



Mixed Variables in Scikit-Learn

What if we have a mix of quantitative and categorical variables, and we only want to dummy encode the categorical ones?

We make a `ColumnTransformer`.

```
from sklearn.compose import make_column_transformer

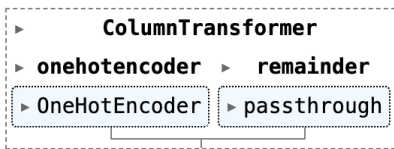
transformer = make_column_transformer(
    (OneHotEncoder(sparse_output=False), ["House Style",
                                          "Neighborhood"]),
    remainder="passthrough")
transformer.fit(df[features])
transformer.transform(df[features])
array([[0., 0., 1., ..., 3., 1., 0.],
       [0., 0., 1., ..., 2., 1., 0.],
       [0., 0., 1., ..., 3., 1., 1.],
       ...,
       [0., 0., 0., ..., 3., 1., 0.],
       [0., 0., 1., ..., 2., 1., 0.],
       [0., 0., 0., ..., 3., 2., 1.]])
```



Visualizing a ColumnTransformer

Scikit-Learn provides a nice visualization of a `ColumnTransformer`.

```
transformer
```



Scaling and Encoding in Scikit-Learn

We can mix scalers and encoders with `ColumnTransformer`!

```
from sklearn.preprocessing import StandardScaler

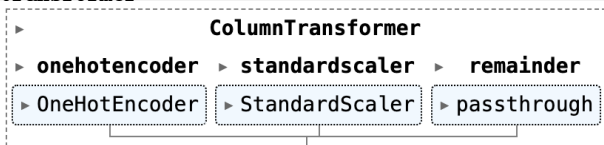
transformer = make_column_transformer(
    (OneHotEncoder(sparse_output=False), ["House Style",
                                          "Neighborhood"]),
    (StandardScaler(), ["Gr Liv Area"]),
    remainder="passthrough")
transformer.fit(df[features])
transformer.transform(df[features])
array([[0., 0., 1., ..., 3., 1., 0.],
       [0., 0., 1., ..., 2., 1., 0.],
       [0., 0., 1., ..., 3., 1., 1.],
       ...,
       [0., 0., 0., ..., 3., 1., 0.],
       [0., 0., 1., ..., 2., 1., 0.],
       [0., 0., 0., ..., 3., 2., 1.]])
```



Scaling and Encoding in Scikit-Learn

Let's visualize this `ColumnTransformer` as well.

```
transformer
```



A Look Ahead

In section tomorrow, you will put all the pieces from the last two lectures together.

- 1 Convert categorical variables to quantitative variables.
- 2 Calculate distances on the transformed data to solve a real problem.

