Lecture 7
Encoding Categorical Variables as Quantitative

Dennis Sun
Stanford University
DATASCI / STATS 112

January 25, 2023

1 Motivation

2 Encoding Categorical Variables

3 Reminders

Review

```
quant_vars = ["Gr Liv Area", "Bedroom AbvGr",
              "Full Bath", "Half Bath"]
df_housing[quant_vars]
```

	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 × 4 columns

We might want to scale the variables first!

Last class, we discussed how to measure the distance between two rows $\vec{x}$ and $\vec{x}'$. For example,

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

What if there are categorical variables?

```
mixed_vars = ["Gr Liv Area", "House Style", "Bedroom AbvGr",
              "Full Bath", "Half Bath"]
df_housing[mixed_vars]
```

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

2930 rows × 5 columns

We have to convert the categorical variables into quantitative variables!

1 Motivation

2 Encoding Categorical Variables

3 Reminders

Encoding Categorical Variables as Quantitative

There is a standard way to encode a categorical variable as a quantitative variable: **dummy encoding** (also called **one-hot encoding**).

	House Style
0	1Story
1	1Story
2	1Story
3	1Story
4	2Story
...	...
2925	SLvl
2926	1Story
2927	SFoyer
2928	1Story
2929	2Story

2930 rows × 1 columns

$\Longrightarrow$

	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	0	0	1	0	0	0	0	0
1	0	0	1	0	0	0	0	0
2	0	0	1	0	0	0	0	0
3	0	0	1	0	0	0	0	0
4	0	0	0	0	0	1	0	0
...	...	...	...	...	...	...	...	...
2925	0	0	0	0	0	0	0	1
2926	0	0	1	0	0	0	0	0
2927	0	0	0	0	0	0	1	0
2928	0	0	1	0	0	0	0	0
2929	0	0	0	0	0	1	0	0

2930 rows × 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

```
pd.get_dummies(df_housing["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	0	0	1	0	0	0	0	0
1	0	0	1	0	0	0	0	0
2	0	0	1	0	0	0	0	0
3	0	0	1	0	0	0	0	0
4	0	0	0	0	0	1	0	0
...	...	...	...	...	...	...	...	...
2925	0	0	0	0	0	0	0	1
2926	0	0	1	0	0	0	0	0
2927	0	0	0	0	0	0	1	0
2928	0	0	1	0	0	0	0	0
2929	0	0	0	0	0	1	0	0

2930 rows × 8 columns

Dummy Encoding in Pandas

You can call `pd.get_dummies` on a `DataFrame`, and it will encode all the categorical variables (leaving the quantitative variables alone).

```
pd.get_dummies(df_housing[mixed_vars])
```

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

2930 rows × 12 columns

Dummy Encoding in Scikit-Learn

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

```
from sklearn.preprocessing import OneHotEncoder

enc = OneHotEncoder()
enc.fit(df_housing[["House Style"]])
output = enc.transform(df_housing[["House Style"]])
output
```

```
<2930x8 sparse matrix of type '<class 'numpy.float64'>'
        with 2930 stored elements in Compressed Sparse Row format>
```

Huh, what's a "sparse matrix"?

An Aside About Sparse Matrices

Notice that the `DataFrame` of dummies consists mostly of 0s. An array that is mostly 0s is called **sparse**.

To store every value in an $m \times n$ array

$$\begin{pmatrix} 0 & 0 & 7 & 0 & \cdots & 0 \\ 0 & -3 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 & \cdots & 0 \end{pmatrix}$$

requires $m \cdot n$ numbers. This is the *dense representation*.

Instead, we can store the k non-zero values and their locations

```
{(0, 2): 7,
 (1, 1): -3,
 ...,
 (m-1, 3): 1}
```

which only requires $3 \cdot k$ numbers. This is the *sparse representation*.

Dummy Encoding in Scikit-Learn

If we want a dense matrix instead of a sparse matrix, set `sparse=False` in `OneHotEncoder`.

```
from sklearn.preprocessing import OneHotEncoder

enc = OneHotEncoder(sparse=False)
enc.fit(df_housing[["House Style"]])
enc.transform(df_housing[["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.]])
```

You can also convert a sparse matrix to dense using `.todense()`:

```
output.todense()
```

Selectively Encoding Variables in Scikit-Learn

What if we have a **`DataFrame`**, and we only want to dummy encode the categorical variables?

We use **`ColumnTransformer`**.

```
from sklearn.compose import ColumnTransformer

enc = ColumnTransformer(
    [("Encoded House Style", OneHotEncoder(), ["House Style"])],
    remainder="passthrough")
enc.fit(df_housing[mixed_vars])
enc.transform(df_housing[mixed_vars])
```

```
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.]])
```

ColumnTransformer in Scikit-Learn

You can mix scalers and encoders with `ColumnTransformer`!

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler

enc = ColumnTransformer(
    [("Scaled Quant Variables", StandardScaler(), ["Gr Liv Area"]),
     ("Encoded House Style", OneHotEncoder(), ["House Style"])],
    remainder="passthrough")
enc.fit(df_housing[mixed_vars])
enc.transform(df_housing[mixed_vars])
```

```
array([[ 0.30926506,  0.,  0., ...,  3., 1.,  0. ],
       [-1.19442705,  0.,  0., ...,  2., 1.,  0. ],
       [-0.33771825,  0.,  0., ...,  3., 1. , 1. ],
       ...,
       [-1.04801492,  0.,  0., ...,  3., 1.,  0. ],
       [-0.21900572,  0.,  0., ...,  2., 1.,  0. ],
       [ 0.9898836 ,  0.,  0., ...,  3., 2.,  1. ]])
```

1 Motivation

2 Encoding Categorical Variables

3 Reminders

Reminders

- Assignment 2 due Friday. Upload to Gradescope before midnight.
- Exam 1 is in class next Monday. Make sure you look at the info and practice on the course website.
- If you have accommodations for Exam 1, you should have received an e-mail from Sophia about when and where to be.