

# Categorical variables

The Classification task introduced us to a type of non-numeric variable called Categorical.

A categorical variable

- Has a value drawn from a discrete set called *Categories* or *Classes*
  - The variable that is the target of a Classification task is Categorical
  - Hence the term "Classification" when the target is categorical

There is **no** ordering relationship between category/class values

- { "Red", "Green", "Blue" } (set notation)
- Versus *ordinal* values [ "Small", "Medium", "Large" ] (sequence notation)

There is no *magnitude* associated with a categorical value

- Even if we could order the colors: how much greater is "Blue" than "Red" ?

We will use  $C$  to denote the set of possible values in a category/class.

Since the values in  $C$  are unordered,  $C$  is mathematically a set of values

$$C = \{c_1, c_2, \dots, \}$$

Since values in a category/class aren't ordered, they are often non-numeric.

Because computers deal with numbers, we will need to *encode* categorical variables as numbers.

In our Titanic example for Binary Classification, there were two obvious categorical variables

- Survived (the target)
- Sex

It might have gone unnoticed that the target was categorical

- Because the values were given to us encoded as numeric 1 (Survived) and 0 (not Survived)

We certainly did notice that Sex was non-numeric

- Because of its encoding as text.

Our point is: don't count on the encoding of raw data in order to determine whether a variable is Categorical

We will illustrate this point with the `Pclass` variable, which has three possible distinct values.

How should we encode a Categorical variable with distinct values from a class  $C$  where  $||C|| > 2$ ?

An obvious choice for such a variable is to encode the values with distinct integers.

This is usually a **bad** idea !

The `Pclass` feature was presented to us encoded as consecutive integers in  $\{1, 2, 3\}$

But it could have just as easily been presented encoded as

- $\{\text{"First"}, \text{"Second"}, \text{"Third"}\}$ .
- or  $\{1, 2, 4\}$

Why is the encoding as  $\{1, 2, 3\}$  any more correct than the encoding as  $\{1, 2, 4\}$ ?



We will give a fuller answer in the module on Model Interpretation. For now:

- In a linear model

$$\hat{\mathbf{y}} = \Theta^T \mathbf{x}$$

- Thus, the contribution of the  $j^{th}$  feature  $\mathbf{x}_j$  to prediction  $\hat{\mathbf{y}}$  is  $\Theta_j * \mathbf{x}_j$
- Consider the encoding of  $\mathbf{x}_j$  ( `Pclass` ) as  $\{1, 2, 3\}$ 
  - The difference in contribution between "First", "Second" and "Third" are all equal
- Consider the encoding of  $\mathbf{x}_j$  ( `Pclass` ) as  $\{1, 2, 4\}$ 
  - The difference in contribution between "Second" and "Third" is twice that of "First" and "Second"

The arbitrary choice of encoding may have an impact on the prediction.

## Bottom line

- Consider whether a feature should be treated as categorical *regardless* of the encoding presented
- Arbitrary mapping of a categorical value to an integer has consequences
  - Avoid it !

We will describe the proper way to encode categorical variables

- And revisit the Titanic example, changing `Pclass` from integer to categorical

# One hot encoding (OHE)

So how should we encode a Categorical variable ?

We can encode a Categorical variable with values in class  $C$  with a vector  $\mathbf{v}$  of length  $||C||$ .

- $\mathbf{v}$  will be all zero *except* at a single index  $j$  where  $\mathbf{v}_j = 1$ .

Consider

$$C = \{c_1, c_2, \dots, c_{||C||}\}$$

The Categorical value  $c_k$  is encoded as vector  $\mathbf{v}$

$$\begin{aligned}\mathbf{v}_j &= 1 && \text{if } j = k \\ \mathbf{v}_j &= 0 && \text{if } j \neq k\end{aligned}$$

That is: position  $k$  of vector  $\mathbf{v}$  is associated with the property: "Is value  $c_k$ "

Hence at most one position of the vector is non-zero.

Let's see what  $\mathbf{v}$  looks like on a number of examples, one for each possible class  $c \in C$ :

	$\mathbf{v}_1$	$\mathbf{v}_2$	$\mathbf{v}_3$	$\dots$	$\mathbf{v}_{  C  }$
$c_1$	1	0	0		0
$c_2$	0	1	0		0
$c_3$	0	0	1		0
$\dots$					
$c_{  C  }$	0	0	0	$\dots$	1

The vector representation essentially replaces the Categorical variable with  $||C||$  binary variables  $\mathbf{v}_1, \dots, \mathbf{v}_{||C||}$

- Each an *indicator* or *dummy* variable:  $\mathbf{v}_k$  indicates whether the value is  $c_k$  or not

This is called the **one hot encoding (OHE)** of a Categorical variable.

- Because at most one indicator in the representation is non-zero

We can use OHE on Categorical variables, whether they be targets or features.

To be concrete let's consider an example  $\mathbf{x}$

- categorical feature  $\mathbf{x}_1$  ("Gender") from categories  $C_{(1)} = \{\text{Male}, \text{Female}\}$
- categorical feature  $\mathbf{x}_2$  ("Location") from categories  $C_{(2)} = \{\text{Urban}, \text{Suburban}, \text{Exurban}\}$

And a few rows from our data set

$$\mathbf{X}' = \begin{pmatrix} \mathbf{const} & \mathbf{Gender} & \mathbf{Location} \\ 1 & \text{Female} & \text{Urban} \\ 1 & \text{Female} & \text{Exurban} \\ 1 & \text{Male} & \text{Exurban} \\ 1 & \text{Male} & \text{Suburban} \\ \vdots & & \end{pmatrix}$$



After our One Hot Encoding we get

$$\mathbf{X}'' = \begin{pmatrix} \text{const} & \text{IsFemale} & \text{IsMale} & \text{IsUrban} & \text{IsSuburban} & \text{IsExurban} \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ \vdots & & & & & \end{pmatrix}$$

---

Notice that the number of features has increased.

Specifically: a single categorical variable  $\mathbf{x}_j$  from a category  $C = \{c_1, c_2, \dots\}$  of size  $\|C\|$

- has been replaced by  $\|C\|$  new binary, indicator variables
  - Is  $c_1$
  - Is  $c_2$
  - $\vdots$
  - Is  $c_{\|C\|}$

# Categorical features versus categorical targets

Although OHE can be applied to features  $\mathbf{x}$  or targets  $\mathbf{y}$ , there are some subtle differences in practice

# Categorical targets

Although we should use OHE to encode the targets, *in practice* you might see targets encoded as integers

- Binary targets as 0/1
- Other targets as integers
  - sklearn method `LabelEncoder` does exactly this

If it's such a bad idea: why does this happen ?

The answer

- It **may** not matter *from a coding perspective*
  - Often, the code need only be able to *distinguish between* target values
    - e.g., restrict the examples to those with a particular value of the target
  - So the encoding of values is not important
  - In fact: `sklearn` is perfectly happy with non-numeric targets for just this reason !

It **may matter** from a mathematical perspective

- Negative/Positive will often be encoded by either 0/1 or  $-1/+1$
- For example: when Negative/Positive encoding is  $-1/+1$

$$10 + \mathbf{y}^{(i)} = 9 \quad \text{for Negative example } i$$

$$10 + \mathbf{y}^{(i)} = 11 \quad \text{for Positive example } i$$

You will often see Categorical values manipulated as mathematical objects when they are used to define Loss Functions.

So please be aware of the purpose for which you are encoding.

## Categorical features

We would love to make the blanket statement: Always use OHE for categorical features.

Unfortunately, there is one model in which OHE may cause a problem

- Linear Regression, with an intercept
- There is a simple fix (i.e., an argument to pass to implementations of OHE)

The issue is called the *Dummy variable* trap and will be explained in a subsequent module.



## Text: another categorical variables

How do you include text variables ? One-hot encoding of the vocabulary !

**Example:** Spam filtering (Classification task with target: Is Spam/Is Not Spam)

In theory, OHE is the solution

- Vocabulary  $V$  of possible words
- $||V||$  indicator variables

In practice

- Vocabulary can be big ! Lots of variables, lots of memory required using OHE
- The representation of a word is "sparse": a single 1 and lots of 0's
  - no relationship between related words: dog, dogs
- Lots of feature engineering possibilities: an ALL CAP feature

We will devote a subsequent module to the topic of Natural Language Processing.

# Recap

- We introduced methods to deal with non-numeric variables
- Unfortunately, there are some nuances

In [4]: `print("Done")`

Done