Categorical Features

How to manage categorical features
(Qualitative data)

Categorical Features Types

Nominal variables are variables that have two or more categories, but which do not have an intrinsic order. For example, a real estate agent could classify their types of property into distinct categories such as houses, condos, co-ops or bungalows.

Dichotomous variables are nominal variables which have only two categories or levels. For example, if we were looking at gender, we would most probably categorize somebody as either "male" or "female".

Ordinal variables are variables that have two or more categories just like nominal variables only the **categories can also be ordered or ranked.** So if you asked someone if they liked the policies of the Democratic Party and they could answer either "Not very much", "They are OK" or "Yes, a lot" then you have an ordinal variable. Why? Because you have 3 categories, namely "Not very much", "They are OK" and "Yes, a lot" and you can rank them from the most positive (Yes, a lot), to the middle response (They are OK), to the least positive (Not very much). However, whilst we can rank the levels, we cannot place a "value" to them; we cannot say that "They are OK" is twice as positive as "Not very much" for example.

Encoding Categorical Data

- Replacing values
- Encoding labels
- One-Hot encoding
- Binary encoding
- Backward difference encoding
- Miscellaneous features

REPLACING VALUE

Replacing the categories with the desired numbers.

This can be achieved with the help of the `replace()` function in `pandas`.

The idea is that you have the liberty to choose whatever numbers you want to assign to the categories according to the business use case.

LABEL ENCODING

It you to convert each value in a column to a number.

Numerical labels are always between 0 and n_categories-1.

You can do label encoding via attributes **`.cat.codes`** on your DataFrame's column.

Label encoding is pretty much intuitive and straight-forward and may give you a good performance from your learning algorithm, but it has as disadvantage that the numerical values can be misinterpreted by the algorithm.

For example : Is vaue x is 2 or 3 times better than value y as ?

ONE HOT ENCODING

The basic strategy is to convert each category value into a new column and assign a `1` or `0` (True/False) value to the column.

This has the benefit of not weighting a value improperly.

There are many libraries out there that support one-hot encoding but the simplest one is using `pandas`' `.get_dummies()` ,SK LEARN preprocessing method.

While one-hot encoding solves the problem of unequal weights given to categories within a feature, it is not very useful when there are many categories, as that will result in formation of as many new columns, which can result in the [curse of dimensionality](#).

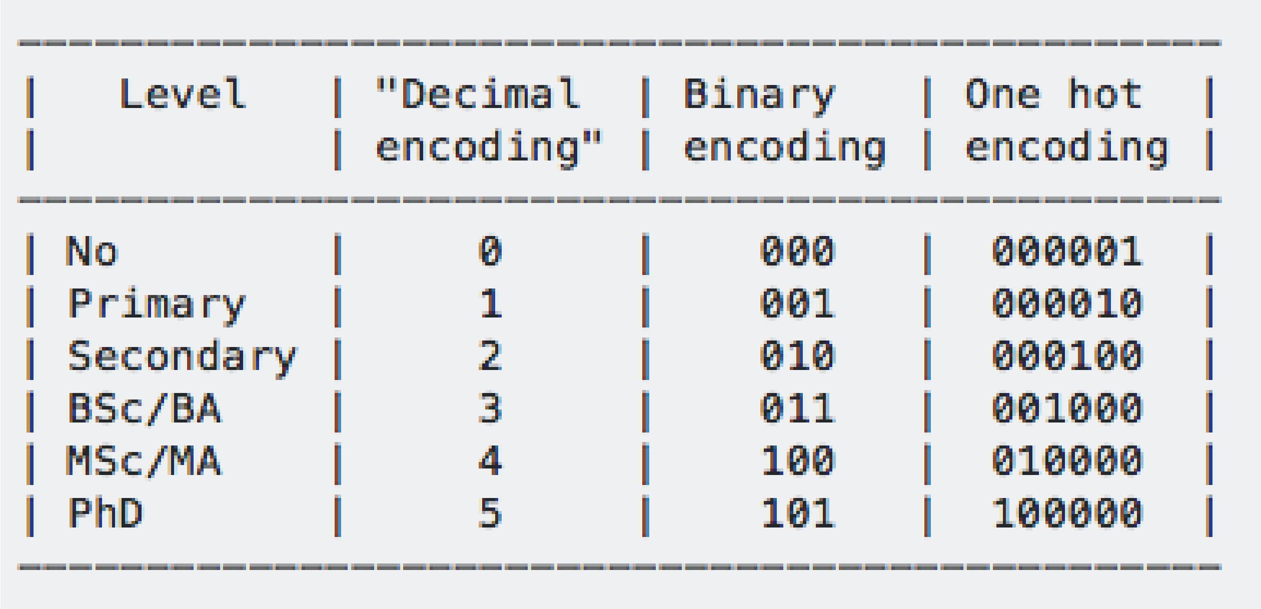
BINARY ENCODING

In this technique, first the categories are encoded as ordinal, then those integers are converted into binary code, then the digits from that binary string are split into separate columns. This encodes the data in fewer dimensions than one-hot.
convert each integer to binary digits.

Each binary digit gets one column. Some info loss but fewer dimensions. Ordinal.

You can do binary encoding via a number of ways but the simplest one is using the `category_encoders` library. You can install category_encoders via **`pip install category_encoders`** on cmd
or just download and extract the .tar.gz file from the site.

ENCODING Reprsentation

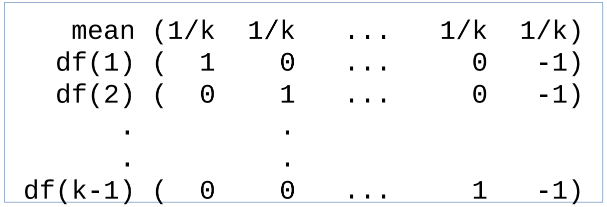


Level	"Decimal encoding"	Binary encoding	One hot encoding
No	0	000	000001
Primary	1	001	000010
Secondary	2	010	000100
BSc/BA	3	011	001000
MSc/MA	4	100	010000
PhD	5	101	100000

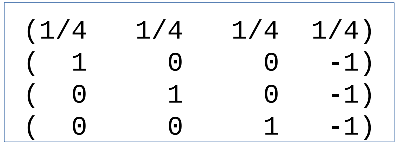
BACKWARD DIFFERENCE

In backward difference coding, the **mean of the dependent variable** for a level is compared with the **mean of the dependent variable for the prior level**.

The interesting thing here is that you can see that the results are not the standard 1’s and 0’s you saw in the dummy encoding examples but rather regressed continuous values.



```
   mean (1/k  1/k   ...   1/k  1/k)
  df(1) (  1    0   ...     0   -1)
  df(2) (  0    1   ...     0   -1)
      .         .
      .         .
df(k-1) (  0    0   ...     1   -1)
```



```
(1/4   1/4   1/4  1/4)
(  1     0     0   -1)
(  0     1     0   -1)
(  0     0     1   -1)
```

Helmert Coding

The mean of the dependent variable for a level is compared to the mean of the dependent variable over all previous levels.

Good for ordinal value.

where **k** is the number of categories of the independent variable. For example, an independent variable with four categories has a Helmert contrast matrix of the following form:

```
   mean (1/k        1/k    ...       1/k       1/k       1/k)
  df(1) (   1   -1/(k-1)   ...  -1/(k-1)  -1/(k-1)  -1/(k-1))
  df(2) (   0          1   ...  -1/(k-2)  -1/(k-2)  -1/(k-2))
      .                .
      .                .
df(k-2) (   0          0   ...         1      -1/2      -1/2)
df(k-1) (   0          0   ...         0         1        -1)
```

```
(1/4   1/4   1/4   1/4)
(  1  -1/3  -1/3  -1/3)
(  0     1  -1/2  -1/2)
(  0     0     1    -1)
```