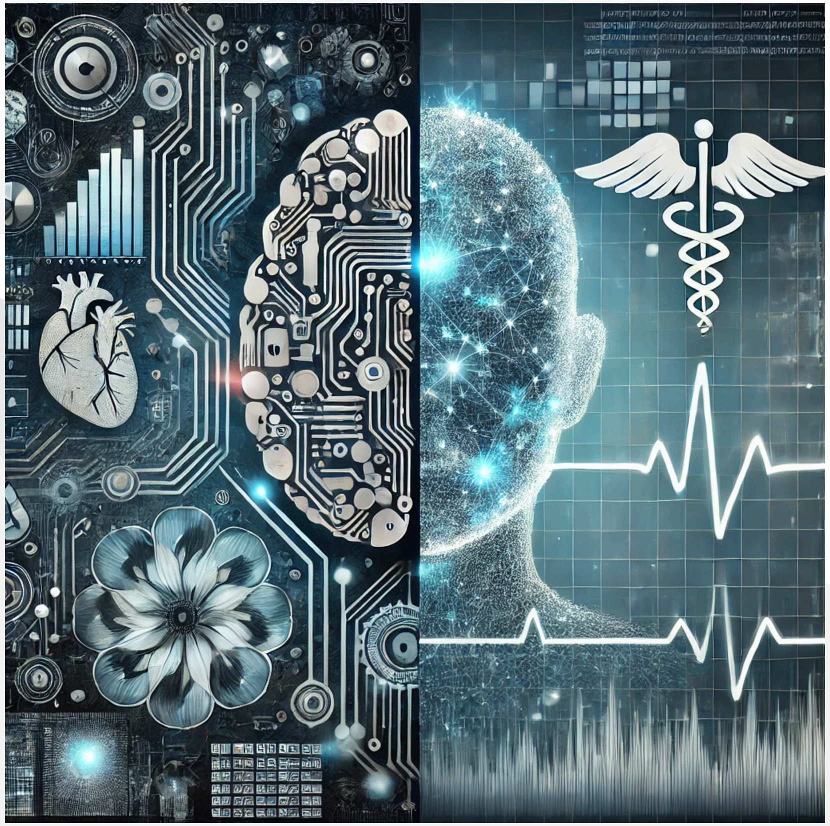
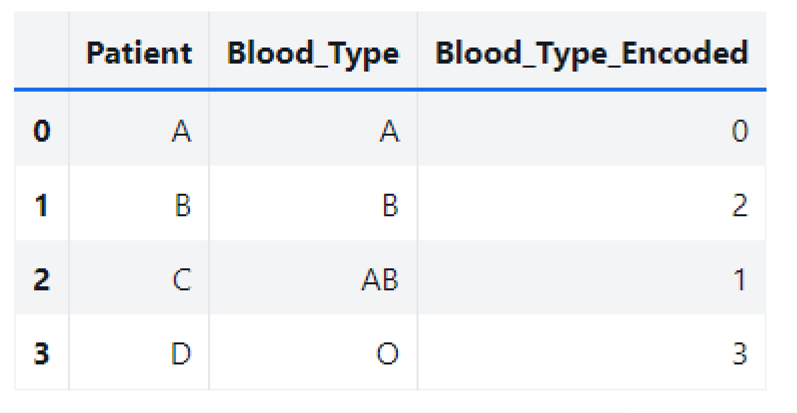


Feature Engineering

Categorical Variables

The Problem with Simple Integer Encoding

In a dataset containing categorical information like "Blood Type", it might seem reasonable to assign integer values:

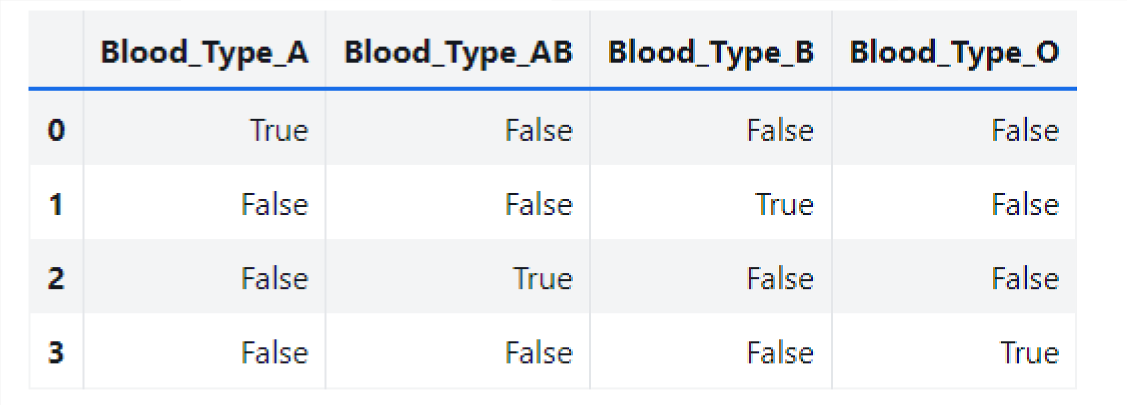


	Patient	Blood_Type	Blood_Type_Encoded
0	A	A	0
1	B	B	2
2	C	AB	1
3	D	O	3

However, this simple encoding creates an unintended ordinal relationship between the categories. This can mislead machine learning models into making inaccurate assumptions about the data.

One-Hot Encoding

One-hot encoding solves the problem of artificial ordering by creating binary (0/1) columns for each category. Each column represents a category, and a value of 1 indicates the presence of that category, while a 0 means absence.



	Blood_Type_A	Blood_Type_AB	Blood_Type_B	Blood_Type_O
0	True	False	False	False
1	False	False	True	False
2	False	True	False	False
3	False	False	False	True

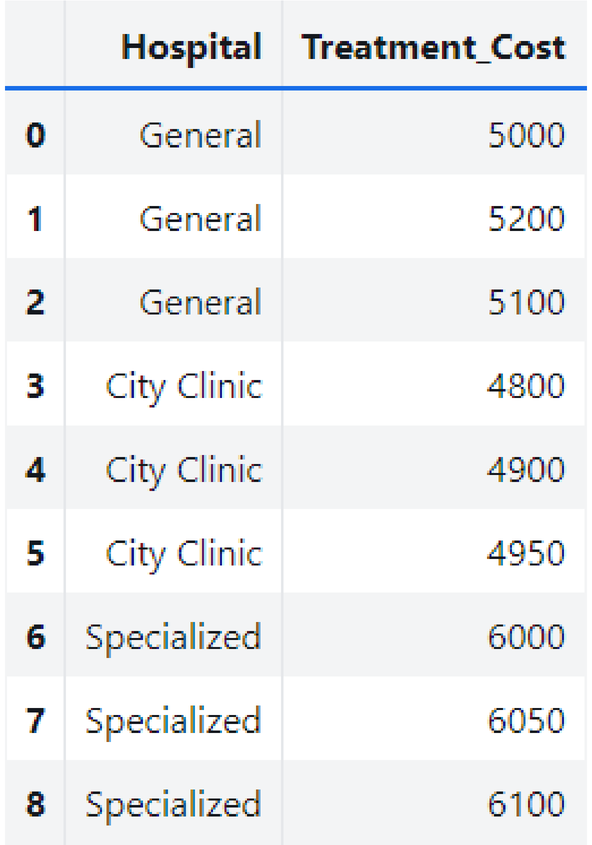
One-Hot Encoding: example

Suppose we are trying to predict the average cost of treatment based on the hospital where the patient receives care.
Let's consider that we have data from three hospitals: "General Hospital," "City Clinic," and "Specialized Care Center."

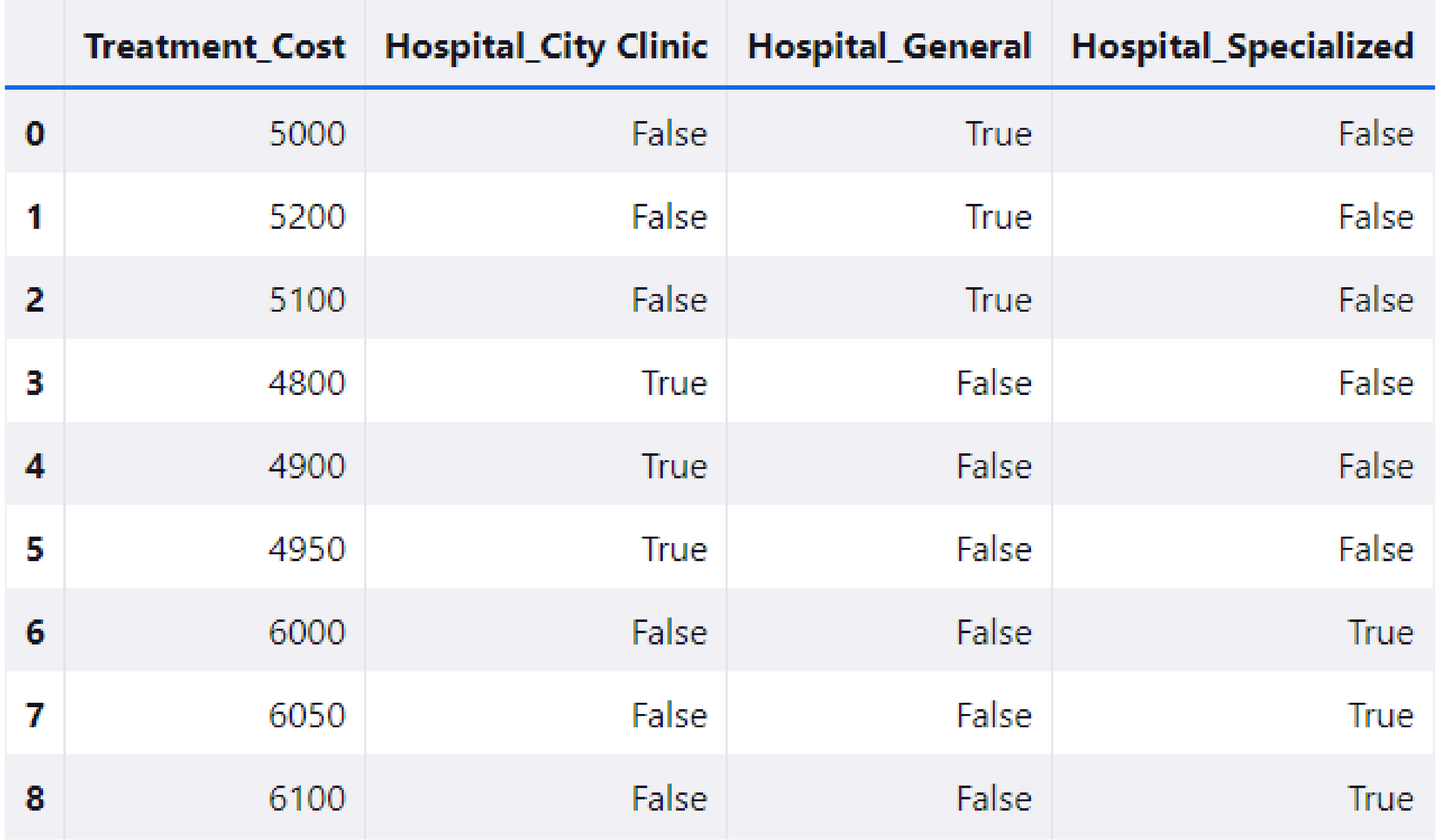
We can train a linear regressor to predict cost of treatment based solely on the identity of the hospital:

$$y = w_1 x_1 + \cdots + w_n x_n + b$$

One-Hot Encoding: example



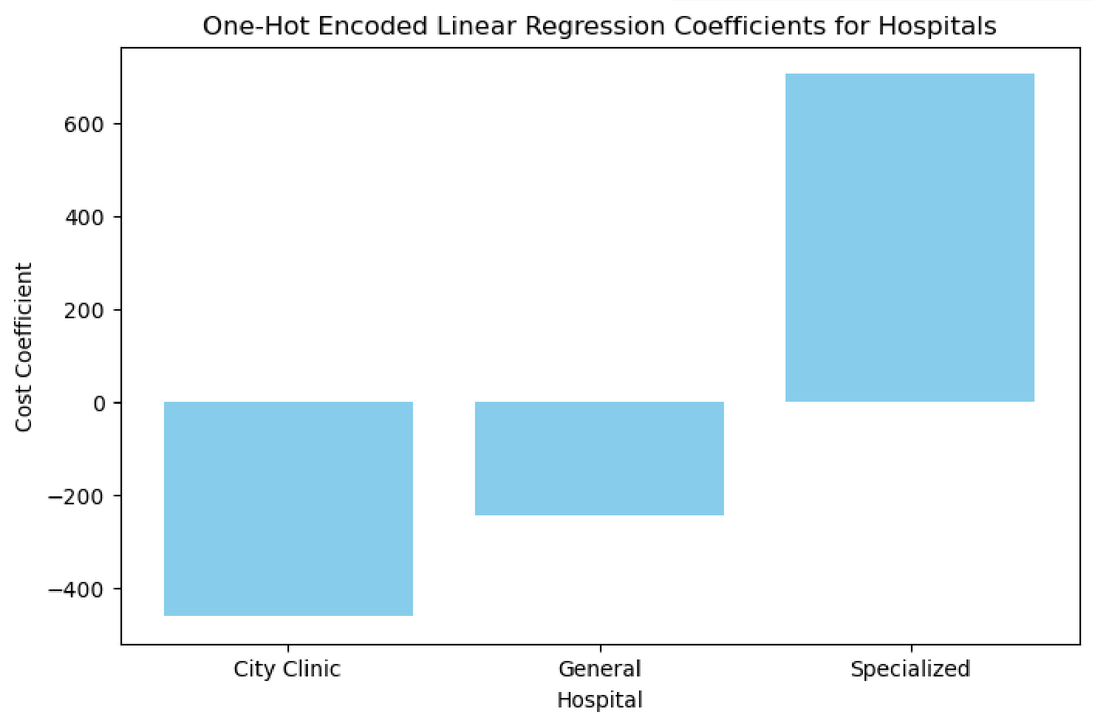
	Hospital	Treatment_Cost
0	General	5000
1	General	5200
2	General	5100
3	City Clinic	4800
4	City Clinic	4900
5	City Clinic	4950
6	Specialized	6000
7	Specialized	6050
8	Specialized	6100



	Treatment_Cost	Hospital_City Clinic	Hospital_General	Hospital_Specialized
0	5000	False	True	False
1	5200	False	True	False
2	5100	False	True	False
3	4800	True	False	False
4	4900	True	False	False
5	4950	True	False	False
6	6000	False	False	True
7	6050	False	False	True
8	6100	False	False	True

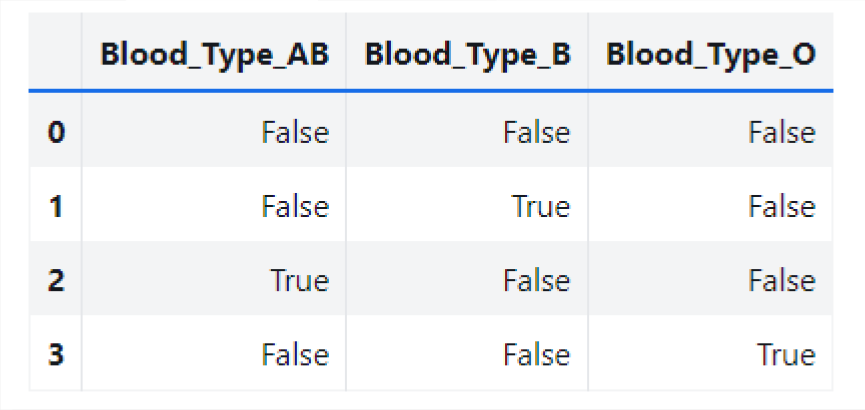
One-Hot Encoding: example

```
# Get the model coefficients and intercept
Coefficients: [-461.11111111 -244.44444444  705.55555556]
Intercept: 5344.444444444444
```



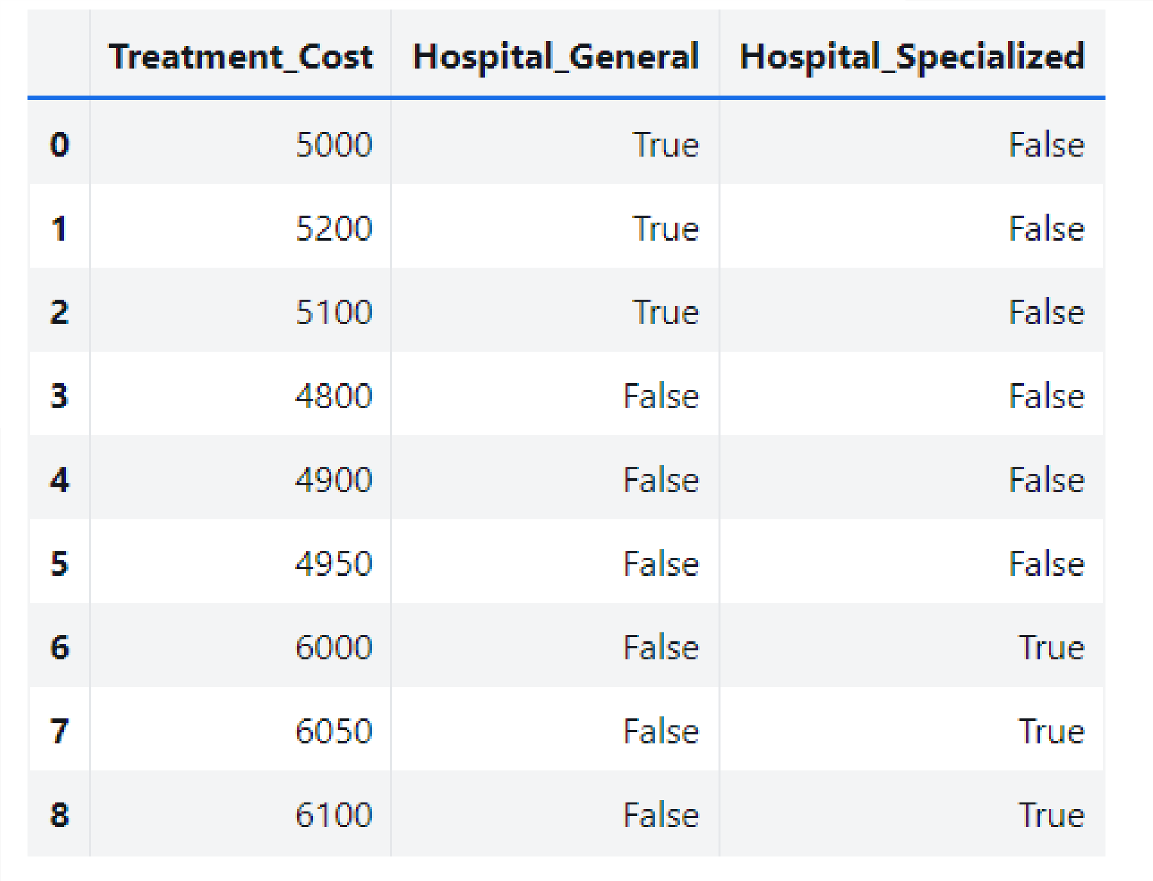
Dummy Coding

Dummy encoding is similar to one-hot encoding, but it drops one of the categories to avoid collinearity. This is particularly useful when using linear models, where collinearity can lead to issues during training. The dropped category is represented by the vector of all zeros and acts as *reference category*.



	Blood_Type_AB	Blood_Type_B	Blood_Type_O
0	False	False	False
1	False	True	False
2	True	False	False
3	False	False	True

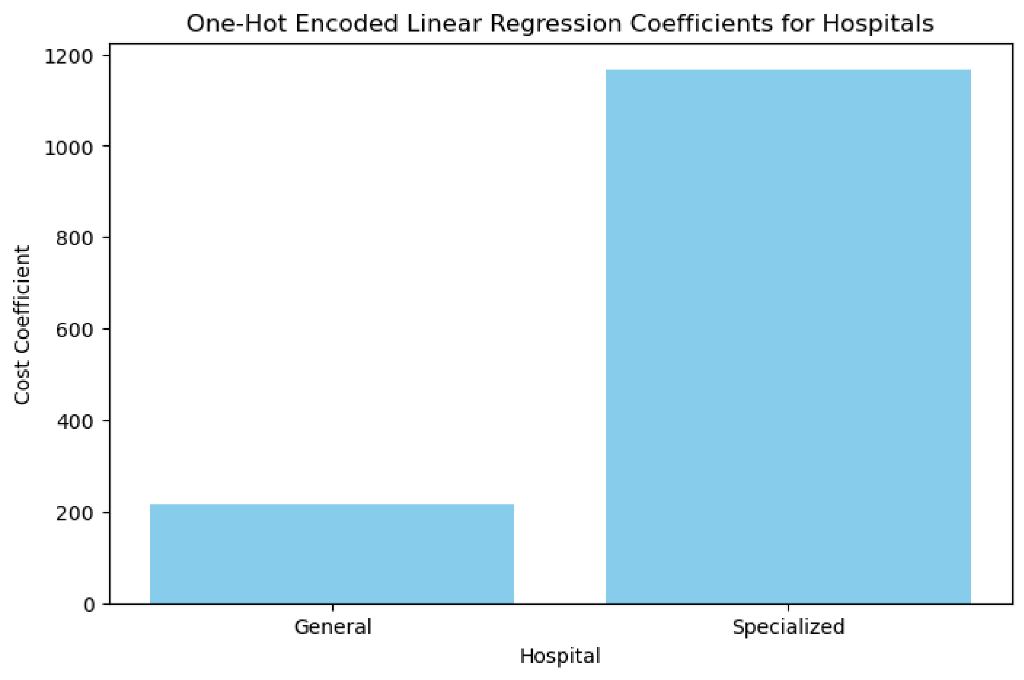
Dummy Encoding: example



	Treatment_Cost	Hospital_General	Hospital_Specialized
0	5000	True	False
1	5200	True	False
2	5100	True	False
3	4800	False	False
4	4900	False	False
5	4950	False	False
6	6000	False	True
7	6050	False	True
8	6100	False	True

Dummy Encoding: example

```
# Get the model coefficients and intercept
Coefficients: [ 216.66666667 1166.66666667]
Intercept: 4883.333333333333
```



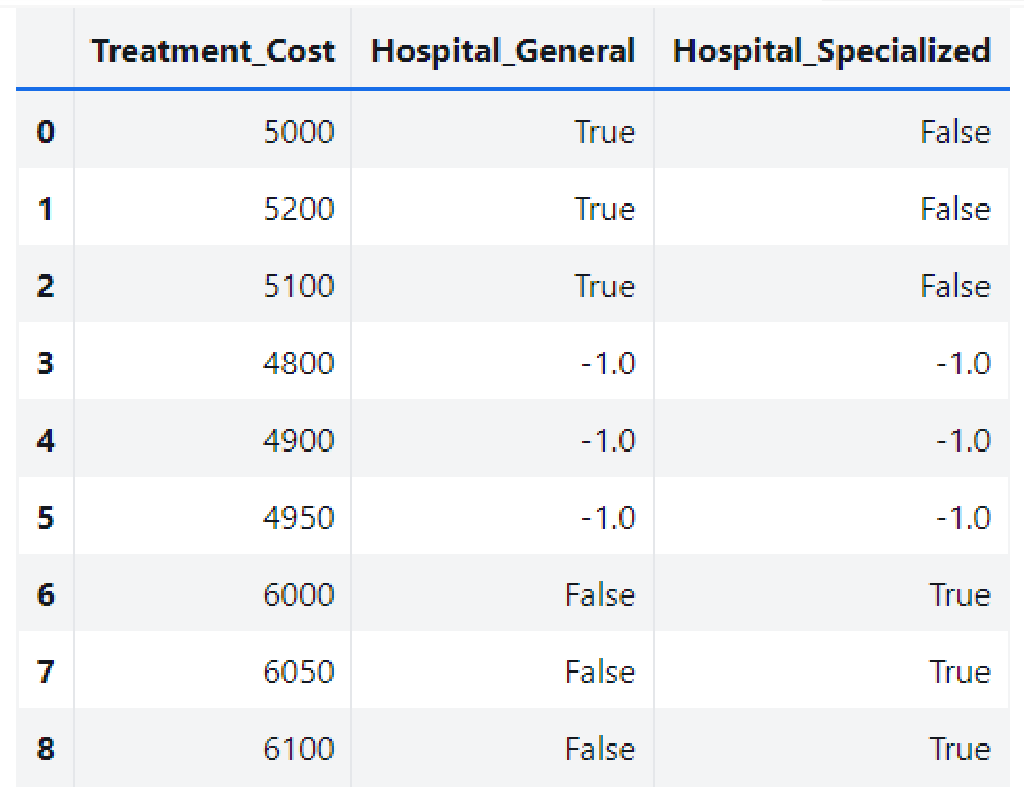
Effect Coding

Effect coding is very similar to dummy coding, with the difference that the reference category is represented by the vector of all –1’s.

Thus, by regressing the effect-coded variables, we compare each level of a categorical variable to the overall (unweighted) mean rather than to a reference category.

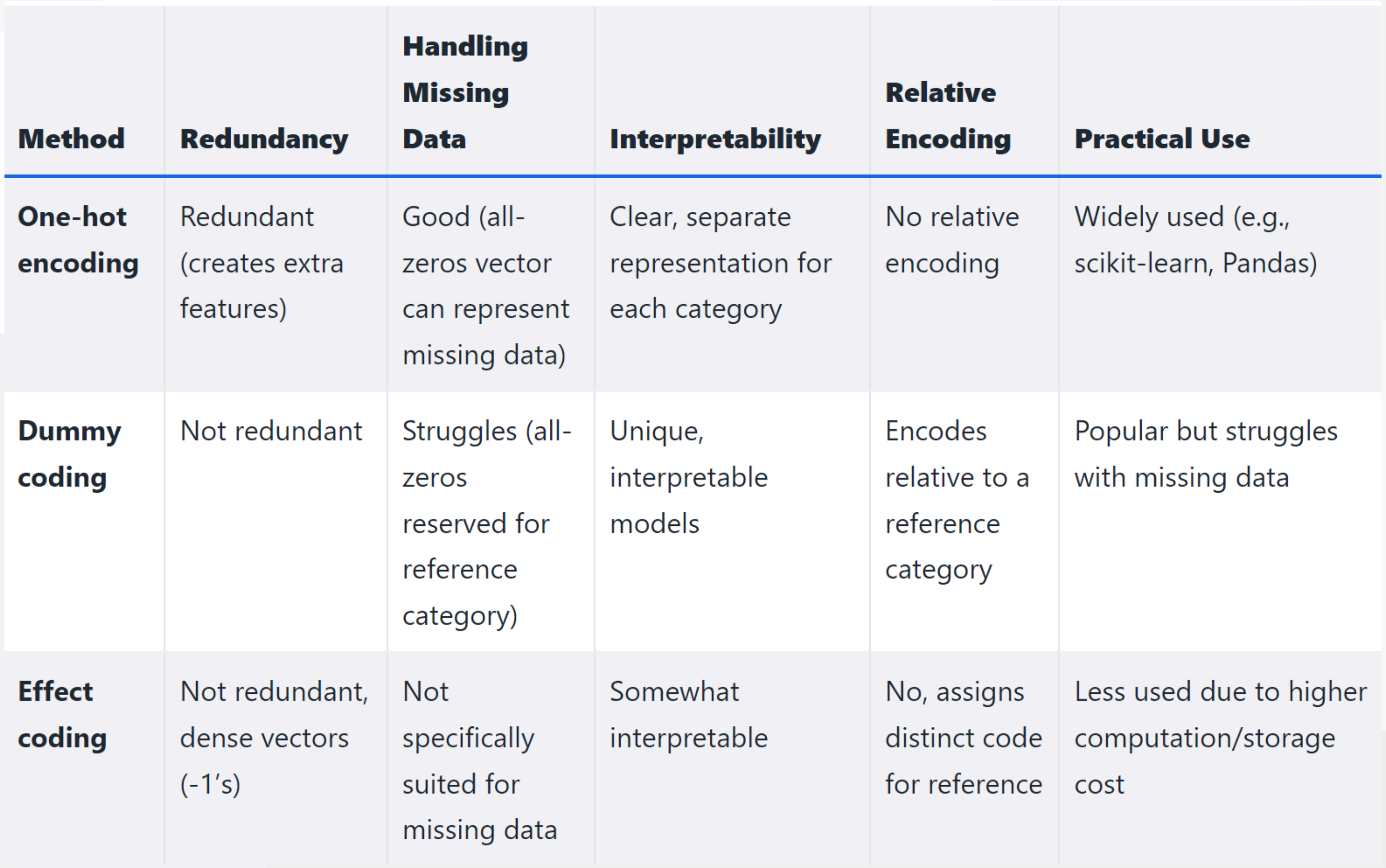
Effect coding can be used in experiments where the goal is to measure the effect of various treatments compared to the average effect across all treatments.

Effect Coding



	Treatment_Cost	Hospital_General	Hospital_Specialized
0	5000	True	False
1	5200	True	False
2	5100	True	False
3	4800	-1.0	-1.0
4	4900	-1.0	-1.0
5	4950	-1.0	-1.0
6	6000	False	True
7	6050	False	True
8	6100	False	True

Pros and Cons of Categorical Variable Encodings



Method	Redundancy	Handling Missing Data	Interpretability	Relative Encoding	Practical Use
One-hot encoding	Redundant (creates extra features)	Good (all-zeros vector can represent missing data)	Clear, separate representation for each category	No relative encoding	Widely used (e.g., scikit-learn, Pandas)
Dummy coding	Not redundant	Struggles (all-zeros reserved for reference category)	Unique, interpretable models	Encodes relative to a reference category	Popular but struggles with missing data
Effect coding	Not redundant, dense vectors (-1’s)	Not specifically suited for missing data	Somewhat interpretable	No, assigns distinct code for reference	Less used due to higher computation/storage cost

Dealing with Large Categorical Variables

When dealing with a large number of unique categories (e.g., thousands of different genetic markers), the encoding strategies mentioned earlier can result in:

- Increased memory consumption.
- Slower computations.
- Potential overfitting due to sparsity.

The challenge is to find a good feature representation that is memory efficient, yet produces accurate models that are fast to train.

Dealing with Large Categorical Variables

Generally, existing solutions can be reduecd to:

- Do nothing fancy with the encoding. Use a **simple model** that is cheap to train. Feed one-hot encoding into a linear model (logistic regression or linear support vector machine) on lots of machines.
- Compress the features. There are two choices:
  - **Feature hashing**, popular with linear models
  - **Bin counting**, popular with linear models as well as trees

Feature Hashing

Feature hashing, also known as the hashing trick, is a technique where each category is hashed into a fixed number of bins. This approach allows for a more compact representation of large categorical variables, but it introduces a risk of collisions where different categories are hashed into the same bin.

$$\phi_i(x) = \sum_{h(x)=i} \mathrm{sign}(h(x))$$

Bin Counting

The concept of bin counting is straightforward yet powerful: instead of using the value of a categorical variable directly as a feature, we calculate the conditional probability of the target variable given that categorical value.

This approach captures the association statistics between the categorical value and the target, assuming all features are independent.

Bin Counting

As an example, bin counting can be when encoding the occurrence of various medical conditions in a hospital's records, bin counting allows rare diagnoses to be weighted more heavily, reflecting their unique medical significance. This method not only reduces dimensionality but also enhances the model’s ability to handle rare events, such as uncommon diseases, effectively.