

Encoders

[ENSF 444](#)

Data Scaling is still part of Preprocessing and so is Encoding

Learn the different ways to Encode string data

Learn how to combine encoding and machine learning algo into one model

Feature Engineering

Why do we want to encode anything in machine learning

Encoding is needed in machine learning because of categorical variables. Some categories may not have a natural numerical representation so they need to be transformed into numerical format so they can be used in machine learning algos.

Here are some reasons:

Algo compatibility: Many machine learning algos require numerical inputs. For that to work, we need to convert the categorical variables to numerical form for the algos to process them

Feature Representation: Categorical variables often contain important information for predictive modeling. By encoding them properly, you preserve this information and allow the algorithm to utilize it effectively during training.

Prevention of Bias: Encoding categorical variables helps prevent bias that could arise if the algorithm interprets categories as ordered when they are not. For example, if you encode categories as 1, 2, and 3, the algorithm might interpret them as having an inherent order, which may not be true.

Improving Performance: Encoding categorical variables properly can lead to better model performance. For example, some encoding techniques can capture ordinal relationships between categories or reduce the dimensionality of the feature space.

Categorical Variables

aka qualitative variable, which is a type of variable that can assume a set number of distinct categories or groups.

each category represents a qualitative characteristic or attribute. the categories are mutually exclusive, so an observation can only belong to one category.

Two types

Nominal

Consist of categories that lack any inherent order such as the color of a car (red,blue,green, etc)

Ordinal

Contain categories that have a natural order or ranking, like education level for example

Here are some examples for Categorical variables

1. the outcome of a 6 sided die
2. demographic attributes such as disease or status
3. Blood type

Numerical labels in categorical variables are identifiers, in example, 6 sided die, the option 6 is not greater than 1, but represents an outcome.

However in ordinal variables, the labels do have an order.

like strongly disagree to to strongly agree.

Methods for Encoding

One Hot Encoding

This is used for Nominal (unordered) values

Ordinal Encoding

This is used for Ordinal (ordered) values

One Hot Encoding

One Hot encoding is a technique to convert CV(categorical Values) into numerical values for ML models

It creates a new column for each of the category and assigns a binary value of 1 or 0 to indicate the presence or absence of that category.

df		df_encoded			
	Fruit		Fruit_Apple	Fruit_Mango	Fruit_Orange
0	Apple	0	1	0	0
1	Mango	1	0	1	0
2	Apple	2	1	0	0
3	Orange	3	0	0	1

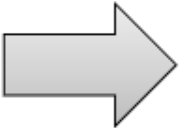
Heres one example of how data can be given numerical values

Ordinal Encoding

Ordinal encoding is transforming categorical values into numerical values by assigning unique integer to each category.

This is useful when the categories have some inherent order or ranking (natural order) like low, medium, high

Ordinal encoding can also reduce the dimensionality of the data and make it easier for algorithms to handle

df				df_encoded	
survey_response				survey_response	
0	strongly disagree			0	0
1	disagree			1	1
2	neutral			2	2
3	agree			3	3
4	strongly agree			4	4
5	strongly agree			5	4

Here is a way to turn data into ordinal

Simple Imputer

Simple Imputer can replace missing values with a constant value or with a statistic such as mean, median, mode calculated from each column.

A SimpleImputer is a preprocessing technique used to handle missing values in a dataset

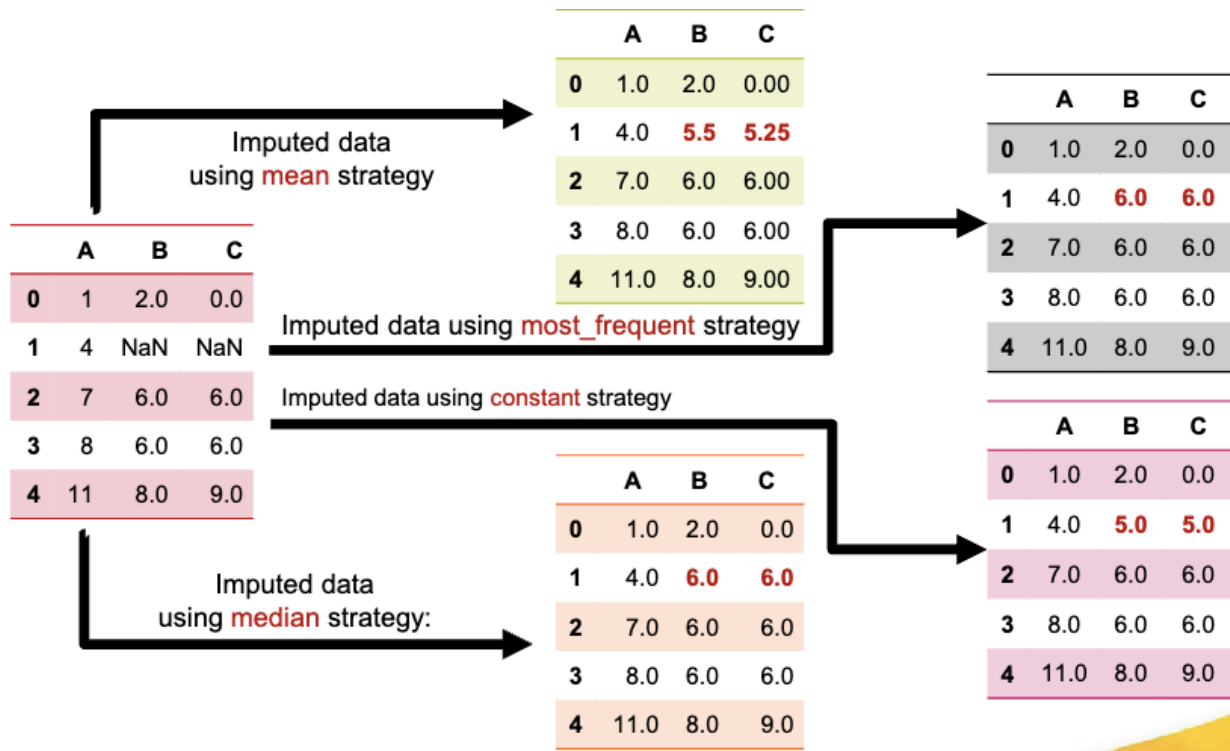
The choice of strategy depends on the type of and distribution of the data, and goal analysis.

Here's how SimpleImputer works:

1. **Initialization:** You create a SimpleImputer object and specify the strategy for imputation. The available strategies are:
 - "mean": Replace missing values with the mean of the non-missing values in the column.
 - "median": Replace missing values with the median of the non-missing values in the column.
 - "most_frequent": Replace missing values with the most frequent value in the column.
 - "constant": Replace missing values with a specified constant value.
2. **Fit:** You then fit the SimpleImputer to your dataset using the `fit()` method. This calculates the imputation statistics (e.g., mean, median, most frequent value) from the non-missing values in each column, depending on the chosen strategy.
3. **Transform:** Finally, you apply the imputation to your dataset using the `transform()` method. This replaces the missing values in each column with the corresponding

imputation statistic calculated during the fitting step.

Here's an example of data being changed



Column Transformer

This class allows you to apply different transformations to different columns in the input data

Really useful since continuous and categorical features need very different kinds of preprocessing

You can use CT object to fit and transform data, just like encoding and scaling

```

import pandas as pd
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder
from sklearn.impute import SimpleImputer

# Define transformers and preprocessor
ordinal_encoder = OrdinalEncoder(categories=[['Low', 'Medium', 'High']])
onehot_encoder = OneHotEncoder(sparse_output = False)
imputer = SimpleImputer(strategy='mean')

# Create a column transformer
preprocessor = ColumnTransformer(
    transformers=[('ordinal', ordinal_encoder, ['Quality']),
                  ('onehot', onehot_encoder, ['Fruit']),
                  ('imputer', imputer, ['Weight'])],
    remainder='passthrough' # Pass through any other columns not specified
)

# Apply the column transformer to the dataset
transformed_data = preprocessor.fit_transform(df)

# Create a dataframe from the encoded data
transformed_df = pd.DataFrame(transformed_data, columns=preprocessor.get_feature_names_out())
# Display the new dataframe
display(transformed_df)

```

	Quality	Fruit	Weight
0	Low	Apple	10.0
1	Medium	Orange	20.0
2	High	Grapes	30.0
3	Low	Pineapple	NaN
4	Medium	Banana	15.0

15

Here is the code for the transformation

Column Transformer - Example

	Quality	Fruit	Weight
0	Low	Apple	10.0
1	Medium	Orange	20.0
2	High	Grapes	30.0
3	Low	Pineapple	NaN
4	Medium	Banana	15.0

	ordinal_Qu ality	onehot_Fru it_Apple	onehot_Fru it_Banana	onehot_Fru it_Grapes	onehot_Fru it_Orange	onehot_Fru it_Pineapple	imputer_W eight
0	0.0	1.0	0.0	0.0	0.0	0.0	10.00
1	1.0	0.0	0.0	0.0	1.0	0.0	20.00
2	2.0	0.0	0.0	1.0	0.0	0.0	30.00
3	0.0	0.0	0.0	0.0	0.0	1.0	18.75
4	1.0	0.0	1.0	0.0	0.0	0.0	15.00

Heres how it would be transformed after