Data Preprocessing :

`Encode categorical variables:`

There are plenty of methods to encode categorical variables into numeric and each method comes with its own advantages and disadvantages.

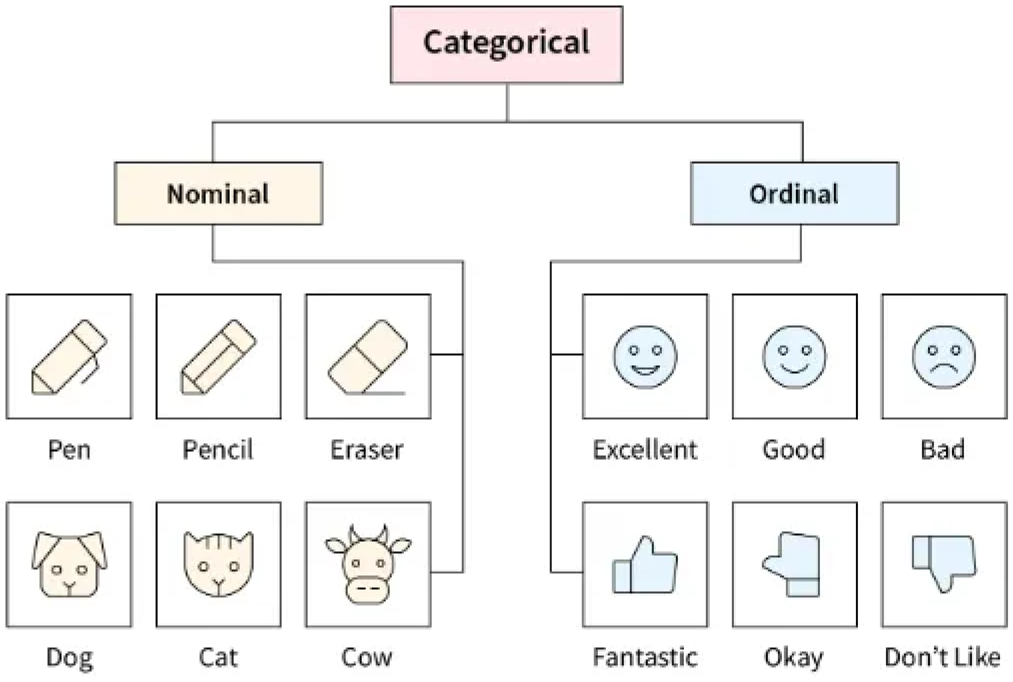
To discover them, we will see the following ways to encode categorical variables:

1. One-hot/dummy encoding
2. Label / Ordinal encoding
3. Target encoding
4. Frequency / count encoding
5. Binary encoding
6. Feature Hashing

In machine learning, there are two main types of categorical data:

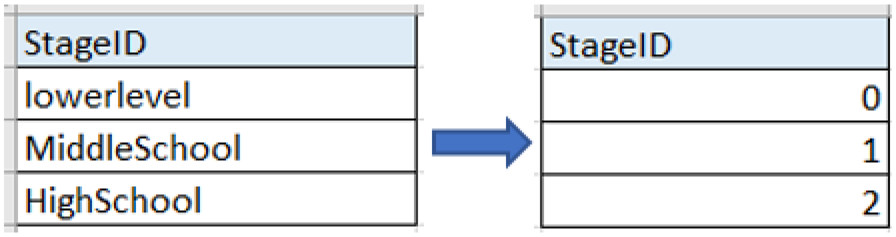
Nominal(label) Categorical Data: Nominal variables represent categories without any specific order or ranking between them. The categories are simply distinct groups. Examples of nominal categorical data include gender (e.g., “Male” and “Female”), country of origin (e.g., “USA,” “UK,” “Germany”), or product categories (e.g., “Electronics,” “Clothing,” “Books”). Nominal variables are often represented using one-hot encoding.

Ordinal Categorical Data: Ordinal variables represent categories that have a natural order or ranking between them. The categories can be ranked based on some criteria or scale. Examples of ordinal categorical data include education level (e.g., “High School,” “Bachelor’s Degree,” “Master’s Degree”), customer satisfaction rating (e.g., “Very Satisfied,” “Satisfied,” “Neutral,” “Dissatisfied,” “Very Dissatisfied”), or economic status (e.g., “Low Income,” “Middle Income,” “High Income”).



Ordinal Encoding:

Ordinal encoding assigns each unique value to a different integer.



StageID
lowerlevel
MiddleSchool
HighSchool

StageID
0
1
2

This approach assumes an ordering of the categories: “ lowerlevel” (0) < “ MiddleSchool” (1) < “ HighSchool” (2).

Example For Limitation of Label Encoding

An attribute having output classes **Mexico, Paris, Dubai**. On Label Encoding, this column lets **Mexico** is replaced with *0*, **Paris** is replaced with *1*, and **Dubai** is replaced with 2.

With this, it can be interpreted that **Dubai** has high priority than **Mexico** and **Paris** while training the model, But actually, there is no such priority relation between these cities here.

One-Hot Encoding:

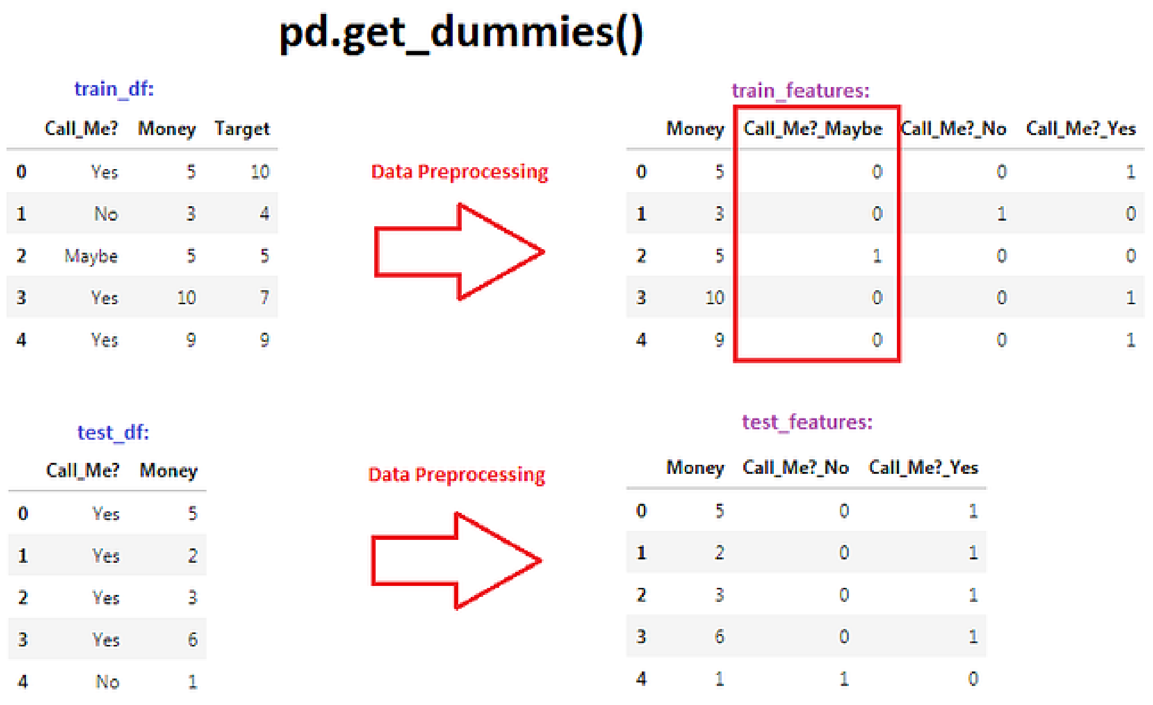
One-Hot Encoding is a technique used to transform categorical variables into a binary representation, where each category is converted into a binary column. It creates new columns for each unique category and assigns a value of 1 or 0 to indicate the presence or absence of that category in a particular row.

Here’s an example code snippet that demonstrates how to perform one-hot encoding using both pandas and scikit-learn libraries in Python

get_dummies (pandas) vs OneHotEncoder (scikit-learn) :

Saving exploded categories is extremely useful when I want to apply the same data pre-processing on my test set. If the total number of unique values in a categorical column is not the same for my train set vs test set, I’m going to have problems.

For example on the training dataframe, when I apply get_dummies() on the ‘Call_Me?’ column; I return 3 new columns because there are 3 unique values. However, on the testing dataframe, I only have 2 unique values under the ‘Call_Me?’ column and therefore get_dummies() returns only 2 new columns.



pd.get_dummies()

train_df:

	Call_Me?	Money	Target
0	Yes	5	10
1	No	3	4
2	Maybe	5	5
3	Yes	10	7
4	Yes	9	9

train_features:

	Money	Call_Me?_Maybe	Call_Me?_No	Call_Me?_Yes
0	5	0	0	1
1	3	0	1	0
2	5	1	0	0
3	10	0	0	1
4	9	0	0	1

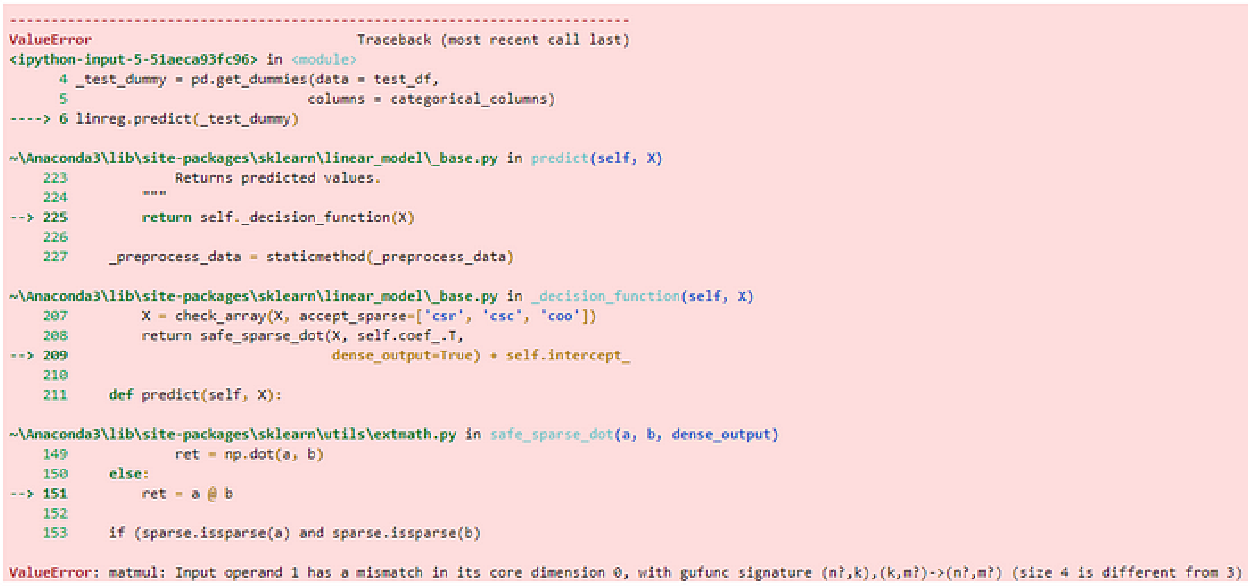
test_df:

	Call_Me?	Money
0	Yes	5
1	Yes	2
2	Yes	3
3	Yes	6
4	No	1

test_features:

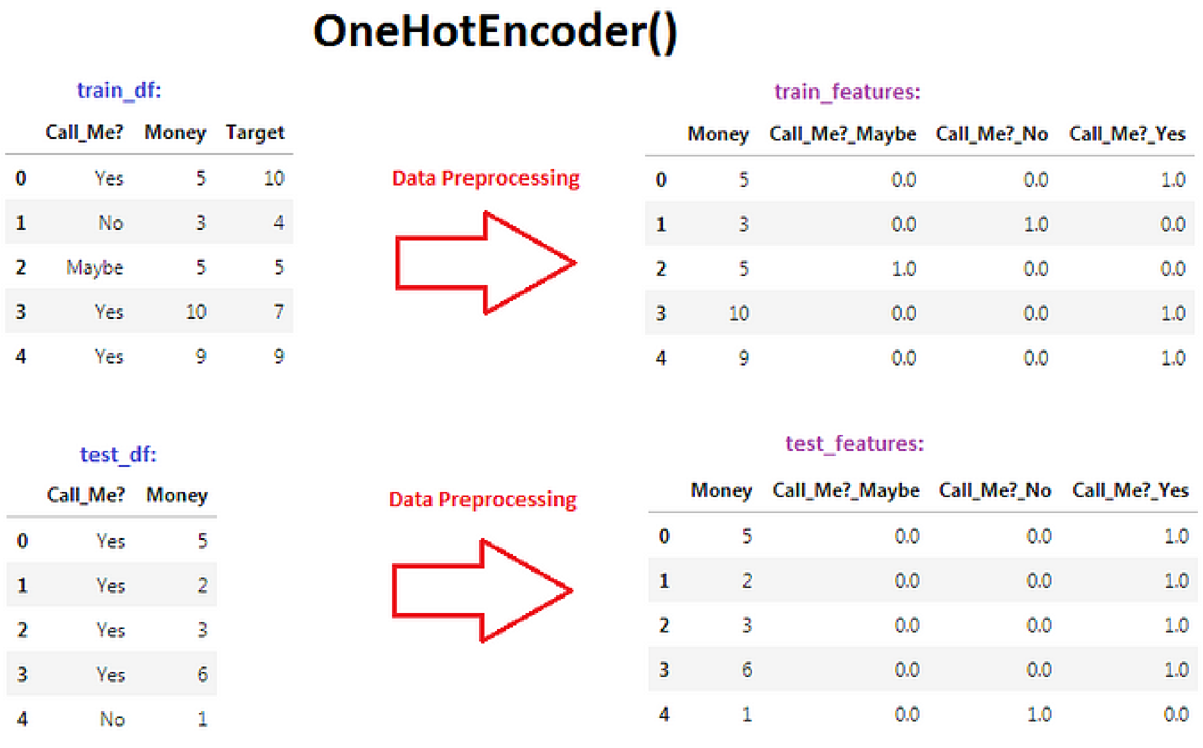
	Money	Call_Me?_No	Call_Me?_Yes
0	5	0	1
1	2	0	1
2	3	0	1
3	6	0	1
4	1	1	0

Moreover, I’ll have errors when I fit a model on the training set and predict on my test features of different shape.



```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-5-51aeca93fc96> in <module>
      4 _test_dummy = pd.get_dummies(data = test_df,
      5                              columns = categorical_columns)
----> 6 linreg.predict(_test_dummy)

~\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py in predict(self, X)
    223             Returns predicted values.
    224         """
--> 225         return self._decision_function(X)
    226 
    227     _preprocess_data = staticmethod(_preprocess_data)

~\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py in _decision_function(self, X)
    207         X = check_array(X, accept_sparse=['csr', 'csc', 'coo'])
    208         return safe_sparse_dot(X, self.coef_.T,
--> 209                                dense_output=True) + self.intercept_
    210 
    211     def predict(self, X):

~\Anaconda3\lib\site-packages\sklearn\utils\extmath.py in safe_sparse_dot(a, b, dense_output)
    149             ret = np.dot(a, b)
    150     else:
--> 151         ret = a @ b
    152 
    153     if (sparse.issparse(a) and sparse.issparse(b)

ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with gufunc signature (n?,k),(k,m?)->(n?,m?) (size 4 is different from 3)
```

OneHotEncoder can transform the test dataframe from the saved exploded categories I fit on the training set.


OneHotEncoder()
Data Preprocessing
Data Preprocessing

train_df:

	Call_Me?	Money	Target
0	Yes	5	10
1	No	3	4
2	Maybe	5	5
3	Yes	10	7
4	Yes	9	9

train_features:

	Money	Call_Me?_Maybe	Call_Me?_No	Call_Me?_Yes
0	5	0.0	0.0	1.0
1	3	0.0	1.0	0.0
2	5	1.0	0.0	0.0
3	10	0.0	0.0	1.0
4	9	0.0	0.0	1.0

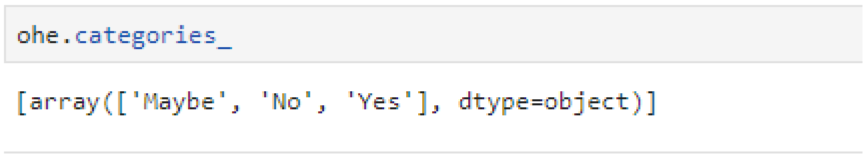
test_df:

	Call_Me?	Money
0	Yes	5
1	Yes	2
2	Yes	3
3	Yes	6
4	No	1

test_features:

	Money	Call_Me?_Maybe	Call_Me?_No	Call_Me?_Yes
0	5	0.0	0.0	1.0
1	2	0.0	0.0	1.0
2	3	0.0	0.0	1.0
3	6	0.0	0.0	1.0
4	1	0.0	1.0	0.0

Moreover, the ohe object has attributes to look for the saved exploded categories.



```
ohe.categories_
```

```
[array(['Maybe', 'No', 'Yes'], dtype=object)]
```

Conclusion

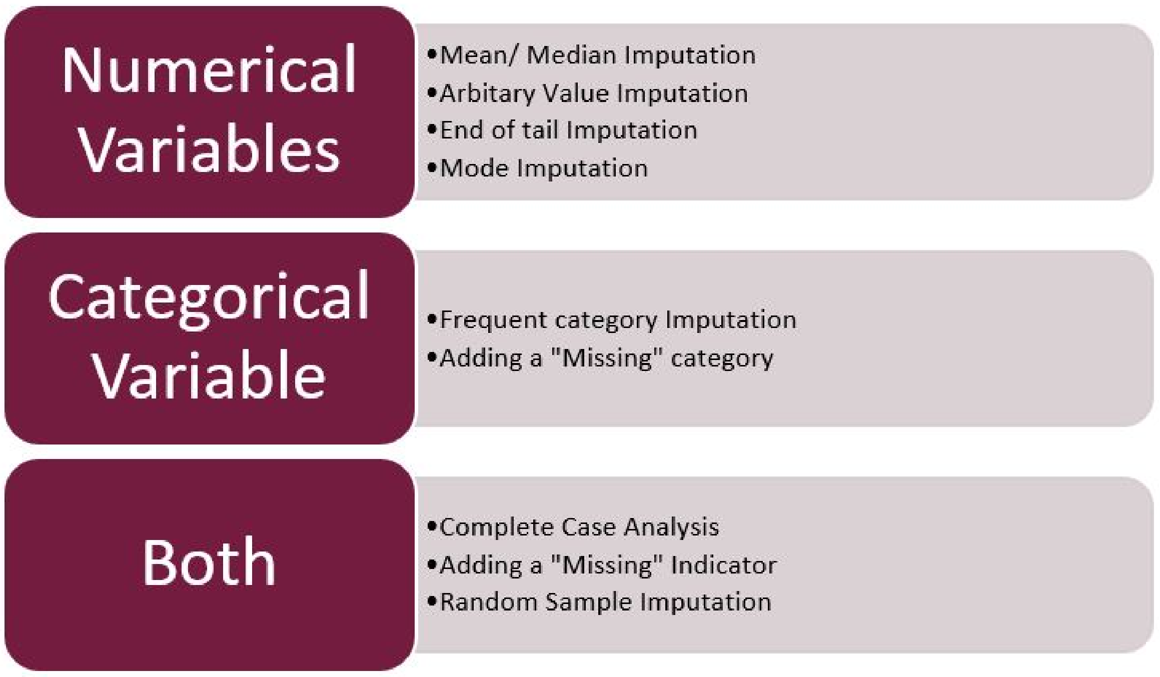
For quick data cleaning and EDA, it makes a lot of sense to use pandas get dummies. However, if I plan to transform a categorical column to multiple binary columns for machine learning, it's better to use OneHotEncoder().

`NaN values Missingvalues :`

Handling missing data: Decide how to deal with missing values, whether by imputation or removal.

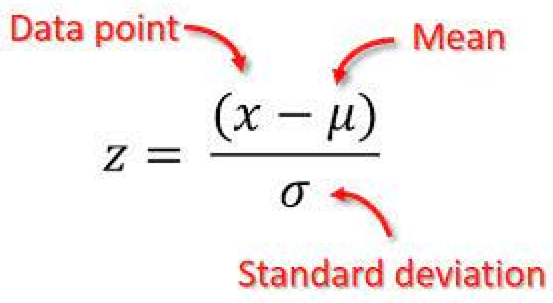
Data Imputation Techniques

Moving on to the main highlight of this article – techniques used in data imputation.



Numerical Variables	•Mean/ Median Imputation •Arbitary Value Imputation •End of tail Imputation •Mode Imputation
Categorical Variable	•Frequent category Imputation •Adding a "Missing" category
Both	•Complete Case Analysis •Adding a "Missing" Indicator •Random Sample Imputation

- **Z-Score:** Calculate the Z-score for each data point. Z-score measures how many standard deviations a data point is from the mean. Typically, a Z-score above a certain threshold (e.g., 2 or 3) indicates an outlier.



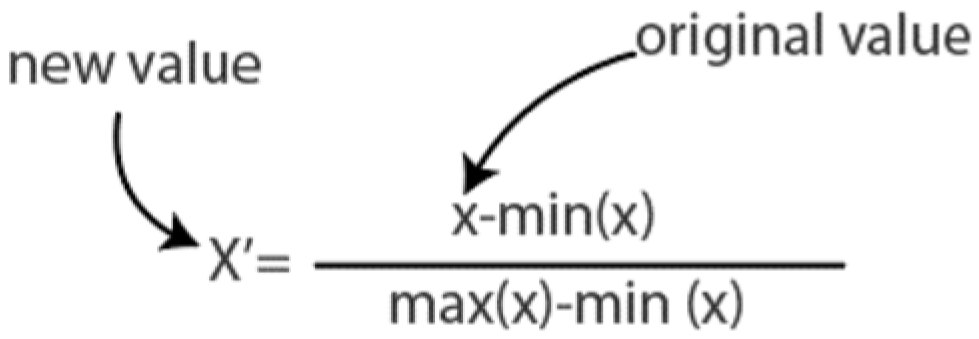
$$z = \frac{(x - \mu)}{\sigma}$$

- **IQR (Interquartile Range):** Calculate the IQR (the range between the first quartile Q1 and the third quartile Q3). Data points beyond a certain range outside of Q1 and Q3 can be considered outliers.

1. **Normalization:**

   - **Objective:** The goal of normalization is to scale the input features to a standard range, typically between 0 and 1.
   - **Method:** For each feature, the minimum value is subtracted, and then the result is divided by the range (difference between the maximum and minimum values).

   Formula:



   - **Benefits:**
     - Ensures that all features have the same scale.
     - Helps the optimization algorithm converge faster.
     - Reduces sensitivity to the scale of input features.

2. **Standardization:**

   - **Objective:** The goal of standardization is to transform the data to have zero mean and a standard deviation of 1.
   - **Method:** For each feature, the mean is subtracted, and then the result is divided by the standard deviation.

   Formula:

$$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation }(x)}$$

- **Benefits:**
  - Centers the data around zero, making it easier to learn the weights.
  - Helps when features have different units or magnitudes.
  - Generally, works well when the data follows a Gaussian distribution.