1.4 - Data Pre-Processing

Giovanni Della Lunga

giovanni.dellalunga@unibo.it

Introduction to Machine Learning for Finance

Bologna - February-March, 2025

Data Pre-Processing

"DATA!
DATA!
DATA!
i can't make
bricks without clay!"
- Sir Arthur Conan Doyle

Introduction

Data Pre-Processing

- Before applying any machine learning model, it is essential to establish a rigorous understanding of **data pre-processing**.
- Poorly prepared data can lead to misleading conclusions, rendering even the most sophisticated algorithms ineffective.
- Data pre-processing introduces several key concepts that are **transversal to all of machine learning**; among these, **handling missing values**, **feature scaling**, **encoding categorical variables**, and **detecting outliers** are crucial.
- These techniques ensure that the input data is structured, consistent, and suitable for training robust models.
- Without proper data preparation, models may struggle with convergence, exhibit bias, or fail to generalize to new data.

Introduction

Data Pre-Processing

In this lesson, we will cover the essential components of data pre-processing, including:

- Handling missing data: imputation and removal strategies
- Feature scaling: normalization and standardization
- Encoding categorical variables
- Detecting and handling outliers
- Feature Selection and Dimensionality Reduction

Data Pre-Processing

- Raw data rarely comes in the form and shape that is necessary for the optimal performance of a learning algorithm.
- On the other hand, the success of a machine learning algorithm highly depends on the quality of the data fed into the model.
- Real-world data is often dirty containing outliers, missing values, wrong data types, irrelevant features, or non-standardized data.
- The presence of any of these will prevent the machine learning model to properly learn.
- For this reason, transforming raw data into a useful format is an essential stage in the machine learning process.

Data Cleaning

- Dealing with inconsistent recording
- Removing unwanted observations
- Removing duplicates
- Investigating outliers
- Dealing with missing items

Subsection 1

Missing Data

Dealing with Missing Data

- The real-world data often has a lot of missing values.
- The cause of missing values can be data corruption or failure to record data.
- The handling of missing data is very important during the preprocessing of the dataset as many machine learning algorithms do not support missing values.

Dealing with Missing Data

Delete Rows/Columns with Missing Values

- One of the easiest ways to deal with missing data is simply to remove the corresponding features (columns) or training examples (rows) from the dataset entirely;
- Missing values can be handled by deleting the rows or columns having null values. If columns have more than half of the rows as null then the entire column can be dropped. The rows which are having one or more columns values as null can also be dropped.
- Remember that, in pandas, rows with missing values can easily be dropped via the **dropna** method.

Dealing with Missing Data

Delete Rows/Columns with Missing Values

Pros:

- A model trained with the removal of all missing values creates a robust model.

Cons:

- Loss of a lot of information.
- Works poorly if the percentage of missing values is excessive in comparison to the complete dataset.

Dealing with Missing Data

Imputing missing values with Mean/Median:

- Columns in the dataset which have numeric continuous values can be replaced with the mean, median, or mode of remaining values in the column.
- This method can prevent the loss of data compared to the earlier method.

Dealing with Missing Data

Imputing missing values with Mean/Median

Pros:

- Prevent data loss which results in deletion of rows or columns
- Works well with a small dataset and is easy to implement.

Cons:

- Works only with numerical continuous variables.
- Can cause data leakage
- Do not factor the covariance between features.

Subsection 2

Categorical Data

A Quick Remind about Data Types

- Machine learning handles various data types: numerical, categorical, ordinal, and textual.
- Proper preprocessing (e.g., encoding, normalization) is essential.
- Choosing the right model depends on the type of data available.

Numerical Data

- **Definition:** Numerical data represents measurable quantities.
- **Types:**
  - **Continuous** - Can take any real value (e.g., height, weight, temperature).
  - **Discrete** - Countable values (e.g., number of transactions, age in years).
- **Example:** House prices, stock prices.

Categorical Data

- **Definition:** Categorical data represents discrete groups or labels.
- **Types:**
  - **Nominal** - No natural order (e.g., colors, gender, country names).
  - **Ordinal** - Has an order but no fixed distance (e.g., education level, customer ratings).
- **Example:** Car brands, job titles.

Textual Data

- **Definition:** Data in the form of text or language-based information.
- **Examples:** Reviews, social media posts, customer feedback.
- **Processing Methods:** Natural Language Processing (NLP), tokenization, embeddings.

What is Categorical Data?

Definitions

- Categorical data refers to data that consists of discrete labels or categories rather than numerical values.
- These categories represent different groups or classifications, such as colors (red, blue, green), gender (male, female, non-binary), or product types (electronics, clothing, furniture).
- There are two types of categorical variables:

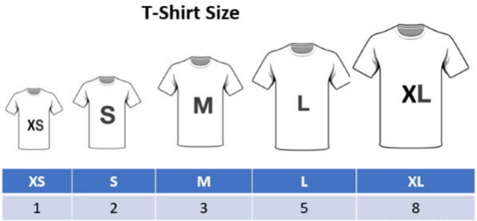
  - **I. Ordinal Variables**

  - **II. Nominal Variables**

Categorical Data: Ordinal Variables

Ordinal Variables

- These variables maintain a natural order in their class of values.
- If we consider the level of education then we can easily sort them according to their education tag in the order of *High School* ¡ *Under-Graduate* ¡ *post-Graduate* ¡ *PhD*.
- The review rating system can also be considered as an ordinal data type where 5 stars is definitely better than 1 star.



XS	S	M	L	XL
1	2	3	5	8

Categorical Data in Credit Scoring

Applicant ID	Employment Type	Credit Rating
001	Salaried	AAA
002	Self-Employed	BBB
003	Unemployed	C
004	Salaried	A

Table: Example of Categorical Data in Credit Scoring

Categorical Data: Nominal Variables

Nominal Variables

- These variables do not maintain any natural/logical order.
- The color of a car can be considered as Nominal Variable as we cannot compare the color with each other.
- It is impossible to state that **Red** is better than **Blu** (subjective!).
- Similarly, Gender is a type of Nominal Variable as again we cannot differentiate between Male, Female, and Others.

Categorical Data: Encoding

- Most of the Machine learning algorithms are designed to work with numeric data.
- Hence, we need to convert Categorical (text) data into numerical data for model building.
- There are multiple encoding techniques to convert categorical data into numerical data. Let us look at some of them.

Categorical Data: Ordinal Encoding

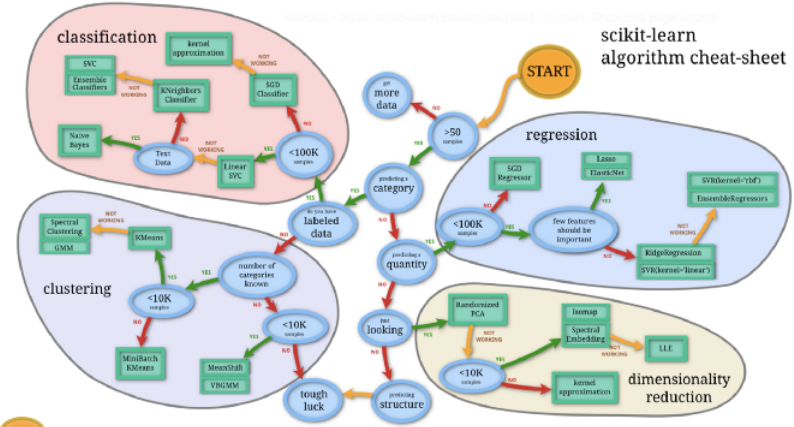
- **Ordinal Encoding** primarily encodes ordinal categories into ordered numerical values.
- Ordinal encoding maps each unique category value to a specific numerical value based on its order or rank.
- Consider for example the credit risk table.
- Here, we can define the ordering of the customer based on their creditworthiness using the OrdinalEncoder methods of
- *OrdinalEncoder(categories=[['AAA', 'AA', 'A', 'B+', ...]])*

Ordinal Encoding - Python Code

Python Implementation:

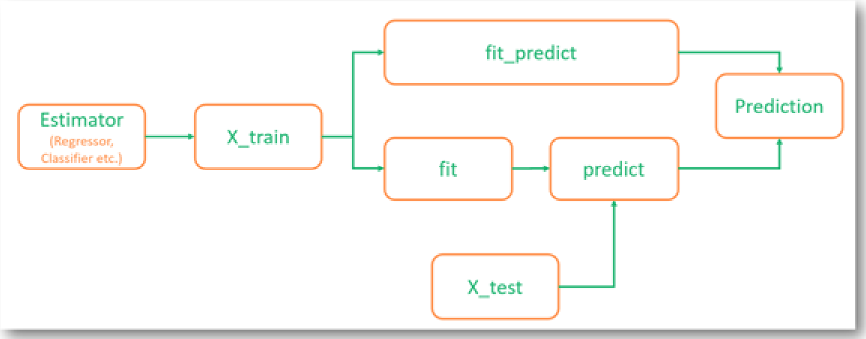
```
from sklearn.preprocessing import OrdinalEncoder

ratings = [["AAA", "AA", "A", "BBB", "BB", "B", "CCC", "CC", "C", "D"]]

le = OrdinalEncoder(categories=ratings)
df["rating_encoded"] = le.fit_transform(df[["rating"]])
df.head()
```

Scikit-learn API design

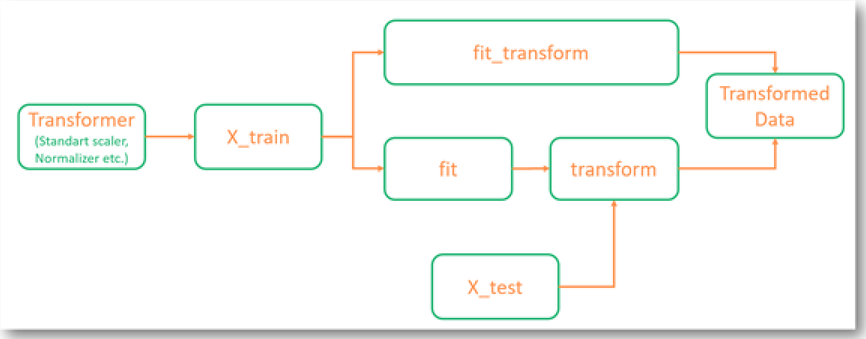


Back

Scikit-learn API design



Scikit-learn API design



Categorical Data: Label Encoding

- The label encoder will convert each category into a unique numerical value.
- If implemented with Sklearn, then this encoder should be used to encode output values, i.e. y, and not the input X.
- It is similar to the ordinal encoder except, here the numeric values are assigned automatically without following any sort of natural order.
- Generally, the alphabetical order of the categorical values is used to determine which numerical value comes first.

Categorical Data: Label Encoding

- Consider for example a target variable 'Job Status' that has four different categories.
- After applying label encoding to this column the four different categories are mapped into integers 0: Full Time, 1: Intern, 2: Part-Time, and 3:Unemployed.
- With this, it can be interpreted that Unemployed have a higher priority than Part-Time, Full Time, and Intern while training the model, whereas, **there is no such priority or relation between these statuses**.
- We cannot define the order of labels with the label encoding technique.

Comparison Between Ordinal and Label Encoding

- Both **Ordinal Encoding** and **Label Encoding** convert categorical data into numbers.
- However, they serve **different purposes** and should not be used interchangeably.

Ordinal Encoding

Comparison Between Ordinal and Label Encoding

Concept:

- Assigns an integer to each category **based on a predefined order**.
- **Preserves order** in the data.

Example: Credit Ratings

Rating	Ordinal Encoding
CCC	0
B	1
BB	2
BBB	3
A	4
AA	5
AAA	6

Label Encoding

Comparison Between Ordinal and Label Encoding

Concept:

- Assigns a unique integer to each category, but **without any ordering**.
- Numbers **do not represent a ranking**.

Example: Car Brands

Car Brand	Label Encoding
Toyota	0
Ford	1
BMW	2
Tesla	3

Label Encoding - Python Code

Comparison Between Ordinal and Label Encoding

Python Implementation:

```
from sklearn.preprocessing import LabelEncoder

data = pd.DataFrame({’car_brand’: [’Toyota’, ’Ford’,
                     ’BMW’, ’Tesla’, ’Ford’]})
encoder = LabelEncoder()
data[’brand_encoded’] = encoder.fit_transform(data[’car_brand’])
print(data)
```

Key Differences

Comparison Between Ordinal and Label Encoding

Comparison Table

Feature	Ordinal Encoding	Label Encoding
Preserves Order?	Yes	No
Use Case	Ordered Categories	Unordered
Assigns Numbers?	Yes	Yes
Represents Ranking?	Yes	No
Risk of Misinterpretation?	If order is wrong	If used on ordered data
Scikit-Learn Class	OrdinalEncoder	LabelEncoder

Conclusions

Comparison Between Ordinal and Label Encoding

- Use **Ordinal Encoding** only when categories have a meaningful order.
- Use **Label Encoding** for nominal data where order does not matter.
- Consider alternative encoding methods (e.g., **One-Hot Encoding**) for categorical variables with many unique values.

Categorical Data: One Hot Encoding

What is One Hot Encoding?

- if there is no ordinal relationship between the categorical variables then ordinal encoding might mislead the model.
- This is because the ordinal encoder will try to force an ordinal relationship on the variables to assume a natural ordering, thus resulting in poor performance.
- In this case, One Hot encoder should be used to treat our categorical variables.
- It will create dummy variables by converting N categories into N features/columns.

Original Binary Variable

Categorical Data: One Hot Encoding

A binary variable example that illustrates one-hot encoding is "Has Mortgage", where applicants either have a mortgage (`Yes`) or do not (`No`).

Applicant ID	Has Mortgage
001	Yes
002	No
003	Yes
004	No

Table: Original Binary Variable

After One-Hot Encoding

Categorical Data: One Hot Encoding

A binary variable example that illustrates one-hot encoding is "Has Mortgage", where applicants either have a mortgage (Yes) or do not (No).

Applicant ID	Has Mortgage (Yes)	Has Mortgage (No)
001	1	0
002	0	1
003	1	0
004	0	1

Table: One-Hot Encoded Binary Variable

After One-Hot Encoding

Categorical Data: One Hot Encoding

We can one-hot encode categorical variables in two ways: by using **get_dummies** in pandas and by using **OneHotEncoder** from sklearn.

Applicant ID	Has Mortgage (Yes)	Has Mortgage (No)
001	1	0
002	0	1
003	1	0
004	0	1

Table: One-Hot Encoded Binary Variable

Subsection 3

Feature Scaling (Normalization)

Feature Scaling

- Before using many ML algorithms (including those for unsupervised learning), it is important to scale feature values so that they are comparable.
- Z-score scaling involves calculating the mean (μ) and SD (σ) from the values of each feature from the training set. Scaled feature values for all data sets are then created by subtracting the mean and dividing by the SD.
- The scaled feature values have a mean of zero and SD of one.

$$\text{scaled feature value} = \frac{V - \mu}{\sigma} \tag{1}$$

Feature Scaling

- Min-max scaling involves calculating the maximum and minimum value of each feature from the training set.
- Scaled feature values for all data sets are then created by subtracting the minimum and dividing by the difference between the maximum and minimum.
- The scaled feature values lie between zero and one:

$$\text{scaled feature value} = \frac{V - V_{min}}{V_{max} - V_{min}} \tag{2}$$