Pandas Data Preprocessing — Data Formatting

What Is Data Preprocessing?

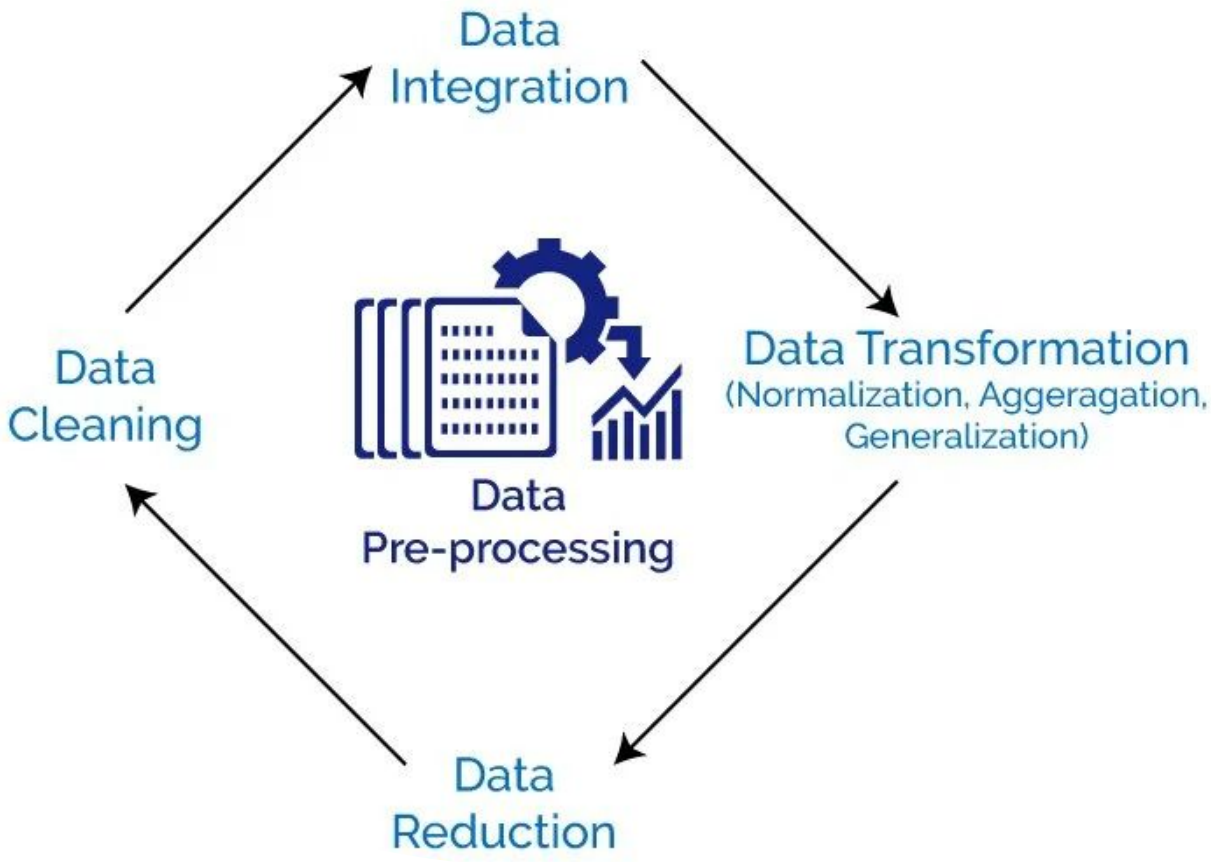
Data preprocessing encompasses several aspects:

---

1. Handling missing values
2. Data formatting
3. Data normalization
4. Data standardization
5. Data binning

Different steps are involved in Data Preprocessing.

- **Data Cleaning**
- **Data Integration**
- **Data Transformation**
- **Data Reduction**



Data Wrangling

- Discovering
- Structuring
- Cleaning
- Enrichment
- Validating

Tasks of Data Wrangling

Discovering:-Firstly, data should be understood thoroughly and examine which approach will best suit

Structuring:- As the data is gathered from different sources, the data will be present in various shapes and sizes. Therefore, there is a need for structuring the data in a proper format.

Cleaning:- Cleaning or removing of data should be performed that can degrade the performance of the analysis.

Enrichment:- Extract new features or data from the given data set to optimize the performance of the applied model.

Validating:- This approach is used for improving the quality of data and consistency rules so that transformations that are applied to the data could be verified.

Publishing:-After completing the steps of Data Wrangling, the steps can be documented so that similar steps can be performed for the same kind of data to save time.

Data formatting

Data formatting is the process of converting data into a common format, facilitating data comparison for users. Unformatted data examples include referencing the same entity with different values within the same column, such as “New York” and “NY.”

Converting Data Types

Firstly, we should ensure that each column is assigned the correct data type. This can be checked using the dtypes attribute:

df.dtypes

```
Tweet Id                              object
Tweet URL                             object
Tweet Posted Time (UTC)               object
Tweet Content                         object
Tweet Type                            object
Client                                object
Retweets Received                      int64
Likes Received                         int64
Tweet Location                        object
Tweet Language                        object
User Id                               object
Name                                  object
Username                              object
User Bio                              object
Verified or Non-Verified              object
Profile URL                           object
Protected or Non-protected            object
User Followers                         int64
User Following                         int64
User Account Creation Date            object
Impressions                            int64
dtype: object
```

function to convert

In this example, we can use the astype() function to convert the “Tweet Location” column to a string, as shown below:

df['Tweet Location'] = df['Tweet Location'].astype('string')

By executing the following statement, all objects can be converted to strings:

obj_columns = df.select_dtypes(include=object).columns.tolist()

df[obj_columns] = df[obj_columns].astype('string')

Other techniques for making categorical data homogeneous include the following aspects:

Remove all whitespace:

df['Tweet Content'] = df['Tweet Content'].str.replace(' ', '')

Remove leading whitespace in strings:

df['Tweet Content'] = df['Tweet Content'].str.lstrip()

Data Homogeneous

This involves both categorical and numerical data. Categorical data should have the same formatting style, such as being in lowercase. To format all categorical data to lowercase, we can use the following statement:

```
df['Tweet Content'] = df['Tweet Content'].str.lower()
```

Cont…

Remove trailing whitespace in strings:

```
df['Tweet Content'] = df['Tweet Content'].str.rstrip()
```

Remove whitespace from both ends:

```
df['Tweet Content'] = df['Tweet Content'].str.strip()
```

Different Values for the Same Feature

The same feature may be represented in different ways. For instance, in our dataset, the column “Twitter Location” contains values like “Columbus,OH” and “Columbus, OH” to describe the same concept. We can use the unique() function to list all unique values in a column:

```
df['Tweet Location'].unique()
```

We can define a function named set_pattern() that searches for specific patterns in a string and performs some replacements within the same string once a pattern is found.

Example- code

```
import re
def set_pattern(x):
    pattern = r'[(A-Z)]\w+,([A-Z])\w+'
    res = re.match(pattern, x)
    if res:
        x = x.replace(',', ', ')
    return x
df['Tweet Location'] = df.apply(lambda x: set_pattern(x['Tweet Location']), axis=1)
print (df)
```

To perform data formatting using Python Pandas. It is recommended to follow these three steps:

- Use the correct format for the data: This is necessary when additional analysis, such as statistical analysis, is required.
- Homogenize the data: This is particularly useful for text data that needs further analysis, such as sentiment analysis.
- Represent the same concept with a single value: This is very useful when data grouping is needed

Normalization or Standardization

Normalization

Normalization works well when the features have different scales and the algorithm being used is sensitive to the scale of the features

Data Normalization is a process of converting a numeric variable into a specified range such as [-1,1], [0,1], etc. A few of the most common approaches to performing normalization are

- Min-Max Normalization, Data Standardization or Data Scaling, etc.
- Single feature scaling transforms each value in a column into a number between 0 and 1. The new values are calculated by dividing the current value by the maximum value of the column.

Single feature scaling transforms

```
import pandas as pd

df = pd.read_csv('datasets/dpc-covid19-ita-regioni.csv')

df.dropna(axis=1, inplace=True)

df.head(7)
```

Single feature scaling transforms each value in a column into a number between 0 and 1. The new values are calculated by dividing the current value by the maximum value of the column.

```
df['tamponi'] = df['tamponi'] / df['tamponi'].max()
print(df)
```

Cont…

Z-score(Normalization)

Z-score transforms each value in a column into a number around 0. The values obtained through z-score transformation typically range between -3 and 3.

The new values are calculated by taking the difference between the current value and the mean and dividing it by the standard deviation. The mean can be obtained using the mean() function, and the standard deviation can be obtained using the std() function. For example, we can calculate the z-scores for the column “deceduti.”

```
df['deceduti'] = (df['deceduti'] - df['deceduti'].mean()) / df['deceduti'].std()

print(df)
```

Log Scaling

Log scaling involves transforming a column to a logarithmic scale. If we want to use the natural logarithm, we can use the log() function from the numpy library. For example, we can apply log scaling to the column.

We use the lambda operator to apply the log function to each element in the column.

```
import numpy as np

df['dimessi_guariti'] = df['dimessi_guariti'].apply(lambda x: np.log(x) if x != 0 else 0)

print(df)
```

Binning

- Data binning or bucketing is a data preprocessing method used to minimize the effects of small observation errors. The original data values are divided into small intervals known as bins and then they are replaced by a general value calculated for that bin.
- Data binning (or bucketing) groups data in bins (or buckets), in the sense that it replaces values contained into a small interval with a single representative value for that interval. Sometimes binning improves accuracy in predictive models.
- Data binning is a type of data preprocessing, a mechanism which includes also dealing with missing values, formatting, normalization and standardization.
- Binning can be applied to convert numeric values to categorical or to sample (quantise) numeric values.
- Binning is a technique for data smoothing. Data smoothing is employed to remove noise from data

1.Equal Frequency Binning:

Data binning, bucketing is a data pre-processing method used to minimize the effects of small observation errors.

The original data values are divided into small intervals known as bins and then they are replaced by a general value calculated for that bin.

This has a smoothing effect on the input data.

There are 2 methods of dividing data into bins:

Equal Frequency Binning: bins have an equal frequency.

Equal Width Binning : bins have equal width with a range of each bin are defined as [min + w], [min + 2w] …. [min + nw] where w = (max – min) / (no of bins).

Equal-Width Binning

Each bin has an equal width, determined by dividing the range of the data into **n** intervals.

Formula:

Each bin has an equal width, determined by dividing the range of the data into nintervals.

Formula:

Bin Width=Max Value–Min ValuenBin Width=nMax Value–Min Value/n

Advantages: Simple to implement and easy to understand.

Disadvantages: May result in bins with highly uneven data distribution.

Advantages: Simple to implement and easy to understand.

Disadvantages: May result in bins with highly uneven data distribution.

Steps in Binning

Sort the Data: Arrange the values of the variable in ascending order.

Define Bin Boundaries: Based on the chosen binning method, determine the intervals.

Assign Data Points to Bins: Allocate each data point to its corresponding bin based on its value.

Applications of Binning

- Equal Frequency Binning: Data values are grouped into bins with approximately the same number of elements.
- Equal Width Binning: Data values are grouped into bins with equal range intervals, regardless of the number of elements in each bin.
- Data Preprocessing: Often used to prepare data for machine learning models by converting continuous variables into categorical ones.
- Anomaly Detection: Helps identify anomalies or outliers by binning data and analyzing the distributions.
- Data Visualization: Used in histograms and bar charts to represent the frequency distribution of data.
- Feature Engineering: Creates categorical features that can enhance the performance of certain machine learning models.