Data Preprocessing

BITS Pilani
Pilani Campus

Data Preprocessing

Overview

In this webinar we’ll learn different data preprocessing techniques using Python.

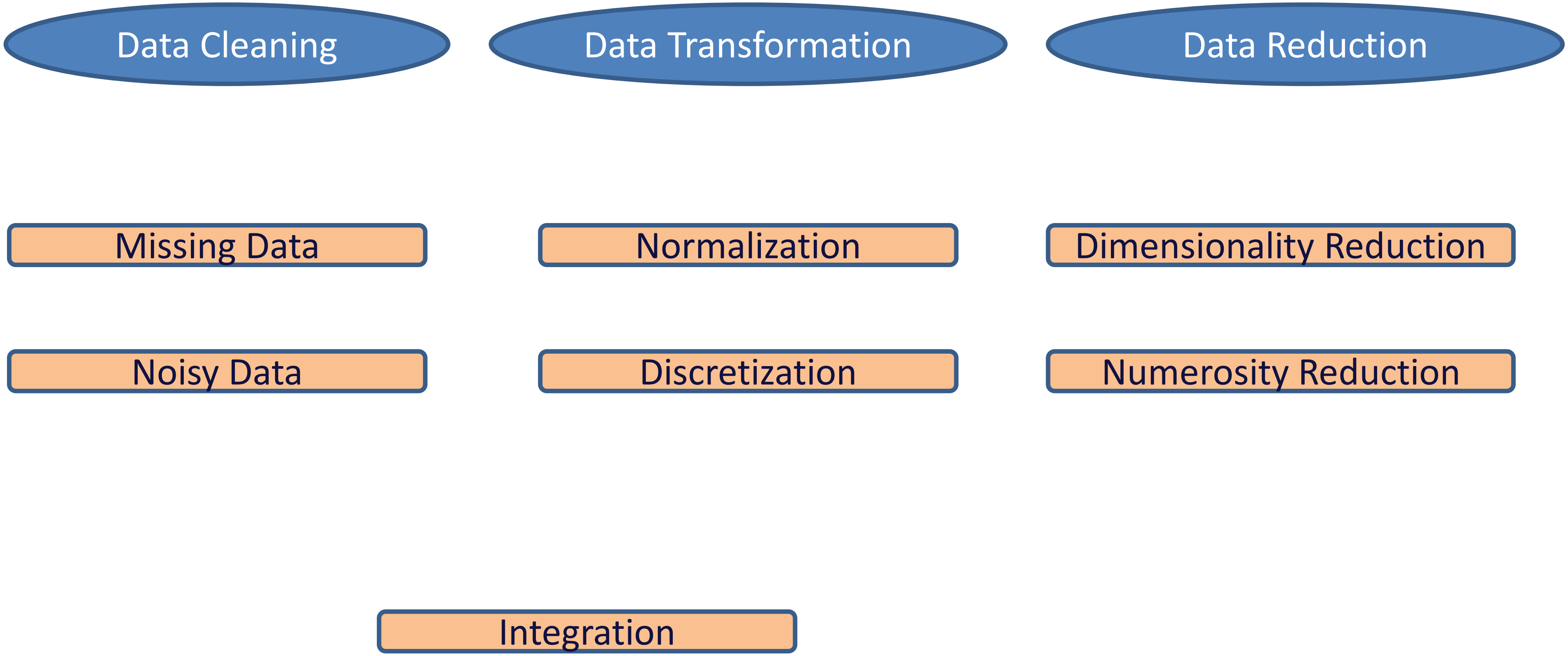
Agenda for Today’s session

- Overview of Python packages for Data Scientists
- What is Data Preprocessing?
- Different techniques for Data Preprocessing
- Implementing Data Preprocessing techniques in python

What is Data Preprocessing ?

- **Data preprocessing** is a **data** mining technique that involves transforming raw **data** into an understandable format.
- Real-world **data** is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors.
- **Data preprocessing** is a proven method of resolving such issues.

Data Preprocessing



Data Cleaning

Data Transformation

Data Reduction

Missing Data

Normalization

Dimensionality Reduction

Noisy Data

Discretization

Numerosity Reduction

Integration

Python packages for Data Scientists

1. Scientifics Computing Libraries : **Pandas**

   Numpy

   Scipy

2. Visualization Libraries : **Matplotlib**

   Seaborn

3. Algorithmic Libraries: **Scikit-learn**

   Statsmodels

Steps in Data Preprocessing

- **Data cleaning**

  Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- **Data integration**

  Integration of multiple databases or files
- **Data transformation**

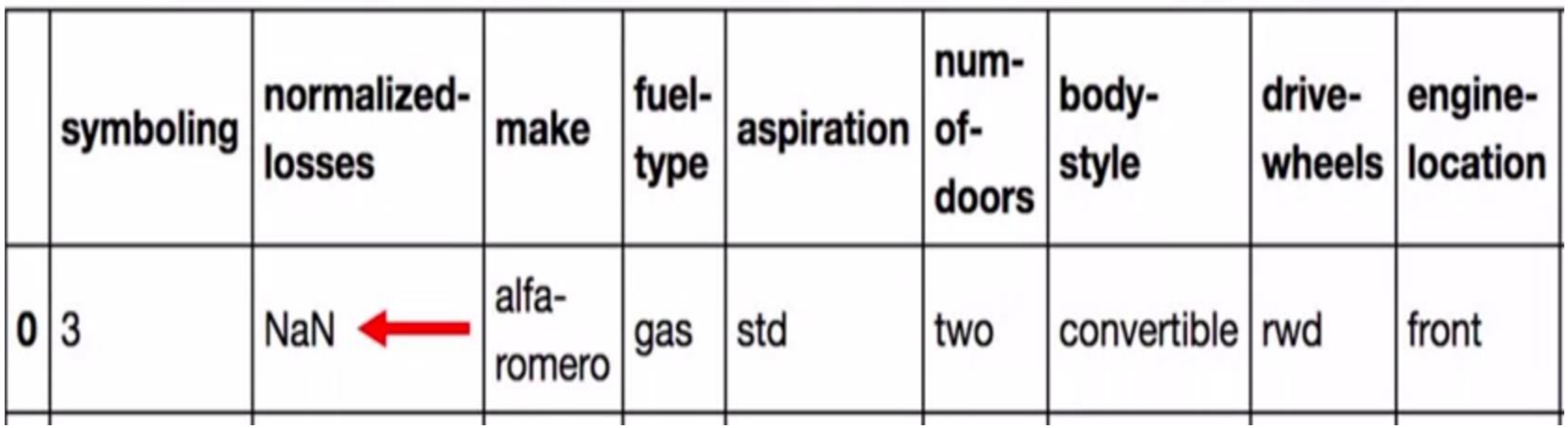
  Normalization and aggregation
- **Data reduction**

  Obtains reduced representation in volume but produces the same or similar analytical results

Data Cleaning: Missing values

- What is missing value?

  When no data value is stored for feature for a particular observation, we say this feature has a missing value.
- Could be represented as ”?”, ”N/A”, O or just a blank cell.



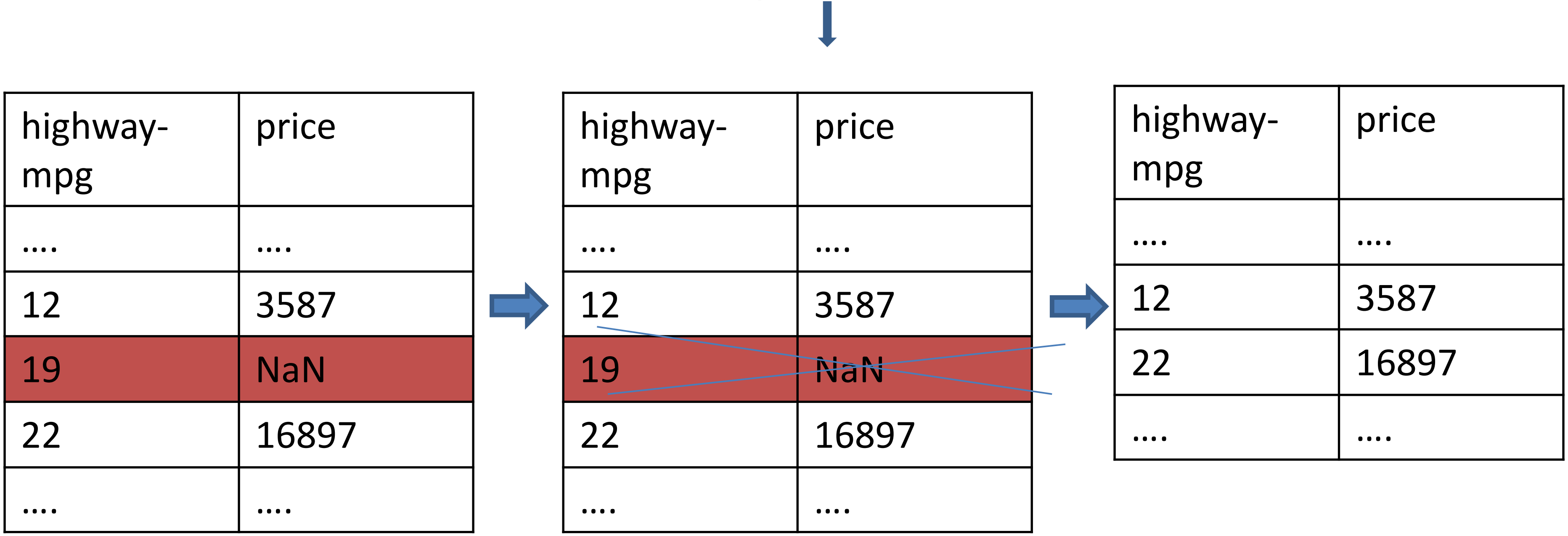
	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front

How to deal with missing data?

- Check with the data collection source
- Drop the missing values
  - drop the variable
  - drop the data entry
- Replace the missing values
  - replace it with an average (or similar data points)
  - replace it by the frequency
- Leave it as missing data

How to drop missing values in Python?

Use **dataframes.dropna():** df.dropna(subset=[“price”], axis =0, inplace=True)



highway-mpg	price
….	….
12	3587
19	NaN
22	16897
….	….

highway-mpg	price
….	….
12	3587
~~19~~	~~NaN~~
22	16897
….	….

highway-mpg	price
….	….
12	3587
22	16897
….	….

axis = 0 **drops the entire row**

axis = 1 **drops the entire column**

How to replace missing values in Python?

Use **dataframe.replace(missing_value, new_value):**

normalized-losses	make
....	....
152	audi
NaN	audi
158	audi
162	audi
....	....

normalized-losses	make
....	....
152	audi
153	audi
158	audi
162	audi
....	....

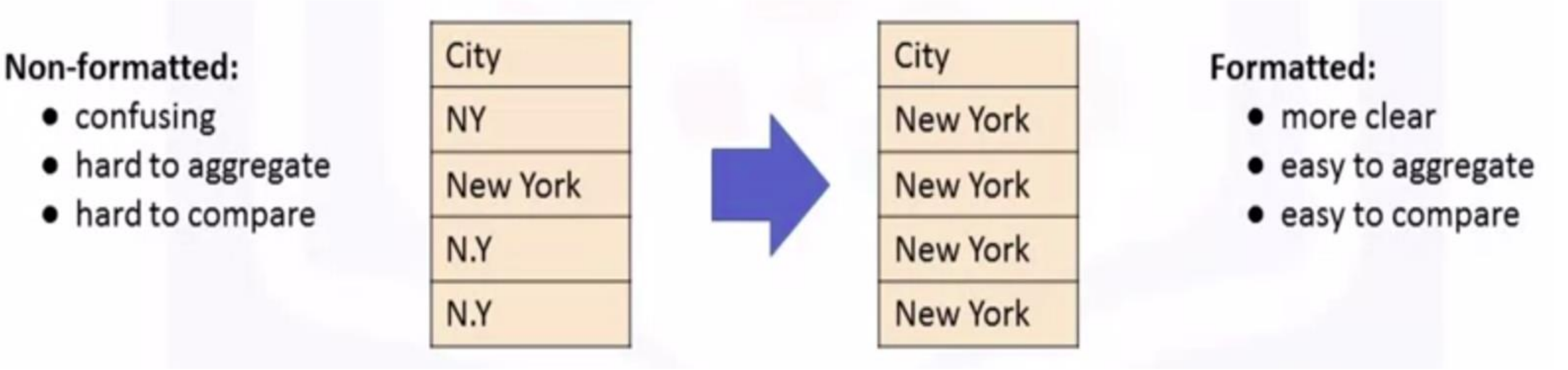
mean = df[“normalized-losses”].mean()

df[“normalized-losses”].replace(np.nan, **mean**)

(missing, new)

Data Integration

- Data are usually collected from different places by different people which may be stored in different formats.
- Bringing data into a common standard of expression that allows users to make meaningful comparisons.



Non-formatted:

- confusing
- hard to aggregate
- hard to compare

City
NY
New York
N.Y
N.Y

City
New York
New York
New York
New York

Formatted:

- more clear
- easy to aggregate
- easy to compare

Data Cleaning: How to Handle Noisy Data?

- **Binning**
  - first sort data and partition into (equal-frequency) bins
  - then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.
- **Regression**
  - smooth by fitting the data into regression functions
- **Clustering**
  - detect and remove outliers

Example

Sorted data for price (in dollars): 4, 8, 9, 15, 21, 21, 24, 25, 26, 28,29, 34

* Partition into equal-frequency (equi-depth) bins:

- Bin 1: 4, 8, 9, 15
- Bin 2: 21, 21, 24, 25
- Bin 3: 26, 28, 29, 34

* Smoothing by bin means:

- Bin 1: 9, 9, 9, 9
- Bin 2: 23, 23, 23, 23
- Bin 3: 29, 29, 29, 29

* Smoothing by bin boundaries:

- Bin 1: 4, 4, 4, 15
- Bin 2: 21, 21, 25, 25
- Bin 3: 26, 26, 26, 34

Data Integration (Example)

- Convert “mpg” to “L/100km” in Car dataset

City-mpg
21
19
….

→

City-L/100km
11.2
12.4
….

```
df[“City-mpg”] = 235/df[“City-mpg”]
Df.rename(columns={“City-mpg”: “City-L/100km},
inplace=True)
```

Data transformation

Uniform features with different range

age	income
10	20000
30	400000
40	60000

Not-Normalized

- “age” and “income” are in different range.
- Hard to compare
- “income” will influence the result more

age	income
0.1	0.04
0.3	0.8
0.4	0.03

Normalized

- Similar value range
- Similar intrinsic influence on analytical model.

Methods for Normalizing data

1. Simple feature scaling

$$x_{new} = \frac{x_{old}}{x_{max}}$$

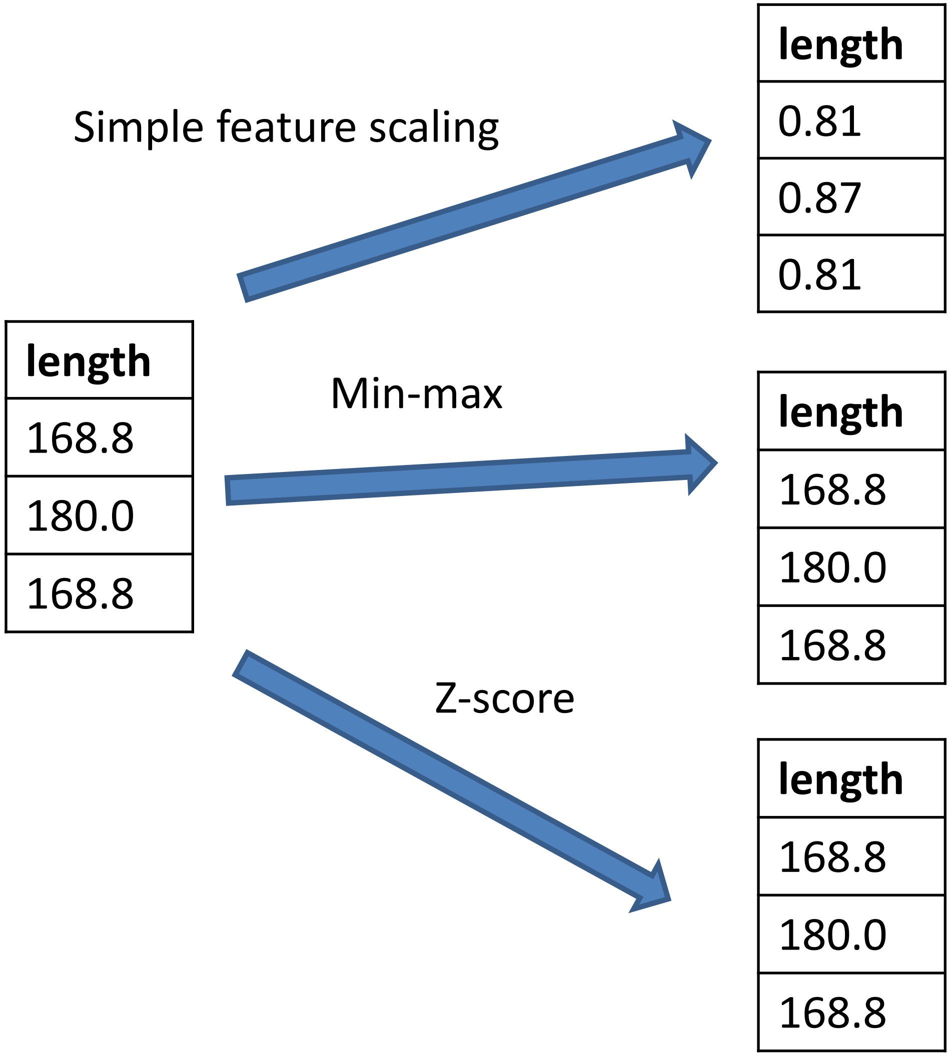
2. Min-Max

$$x_{new} = \frac{x_{old} - x_{min}}{x_{max} - x_{min}}$$

3. Z-Score

$$x_{new} = \frac{x_{old - \mu}}{sigma}$$

Example



length
168.8
180.0
168.8

Simple feature scaling

length
0.81
0.87
0.81

df[“length”] = df[“length”] / df[“length”].max()

Min-max

length
168.8
180.0
168.8

df[“length”] = (df[“length”] - df[“length”].min()) / (df[“length”].max() – df[“length”].min())

Z-score

length
168.8
180.0
168.8

df[“length”] = (df[“length”] - df[“length”].mean()) / (df[“length”].std()

Turning categorical values to numerical variables

- Most statistical models cannot take in objects or strings as inputs.

Solution:

- Add dummy variable for each unique category
- Add 0 or 1 for each category

Car	Fuel
A	Gas
B	Diesel
C	Gas
D	gas

Car	Fuel		Gas	Diesel
A	Gas	….	1	0
B	Diesel	….	0	1
C	Gas	….	1	0
D	gas	….	1	0

“one-hot encoding”

- Use **pandas.get_dummies()** method
- Convert categorical variables to dummy variables (0 or 1)

Fuel
Gas
Diesel
Gas
gas

Gas	Diesel
1	0
0	1
1	0
1	0

pd.get_dummies(df[“Fuel”])

Thank you