PYTHON

HANDLING MISSING DATA

ANITA MILA OKTAFANI

DQLab Live Class Data Analyst with SQL & Python in Google Platform

TABLE OF CONTENT

MEASURES OF CENTER TENDENCY

- Mean
- Median
- Modus

DATA IMPUTATION

MEASURES OF CENTRAL TENDENCY

- Measures of central tendency are values that used to describe a set of data by identifying the center of the data set.
- The most frequently used measures of data centering are the mean, median, and mode.

MEASURES OF CENTRAL TENDENCY: MEAN

- The average or mean is the quotient between the number (sum) of values divided by the number of values. For example:

  50, 70, 90, 60, 50, 65, 100, 70, 70, 55, 90

  value of the mean is

  (50 + 70 + 90 + 60 + 50 + 65 + 100 + 70 + 70 + 55 + 90) / 11 = 70

- To calculate the mean of a column in a dataframe, you can use

```
df['column_name'] . mean()
```

MEASURES OF CENTRAL TENDENCY: MEDIAN

- The median is a value that is located in the middle of a group of data that has been sorted from smallest to largest value or vice versa.

  For an example of calculating the median:

  50, 70, 90, 60, 50, 65, 100, 70, 70, 55, 90

  After sorting, the numbers become:

  50, 50, 55, 60, 65, 70, 70, 70, 90, 90, 100

  Obtained median is 70 (6th data)
- If the amount of data is even then the median value is the average of the two numbers in the middle
- To calculate the median of a column in a dataframe, you can use

```
df['column_name'] . median()
```

MEASURES OF CENTRAL TENDENCY: MODUS

- The mode is the data or value that appears most often

  For example:

  50, 50, 55, 60, 65, 70, 70, 70, 90, 90, 100

  is obtained mode value 70 (appears three times)

- To calculate the mode of a column in a dataframe, you can use

```
df['column_name'] . mode()
```

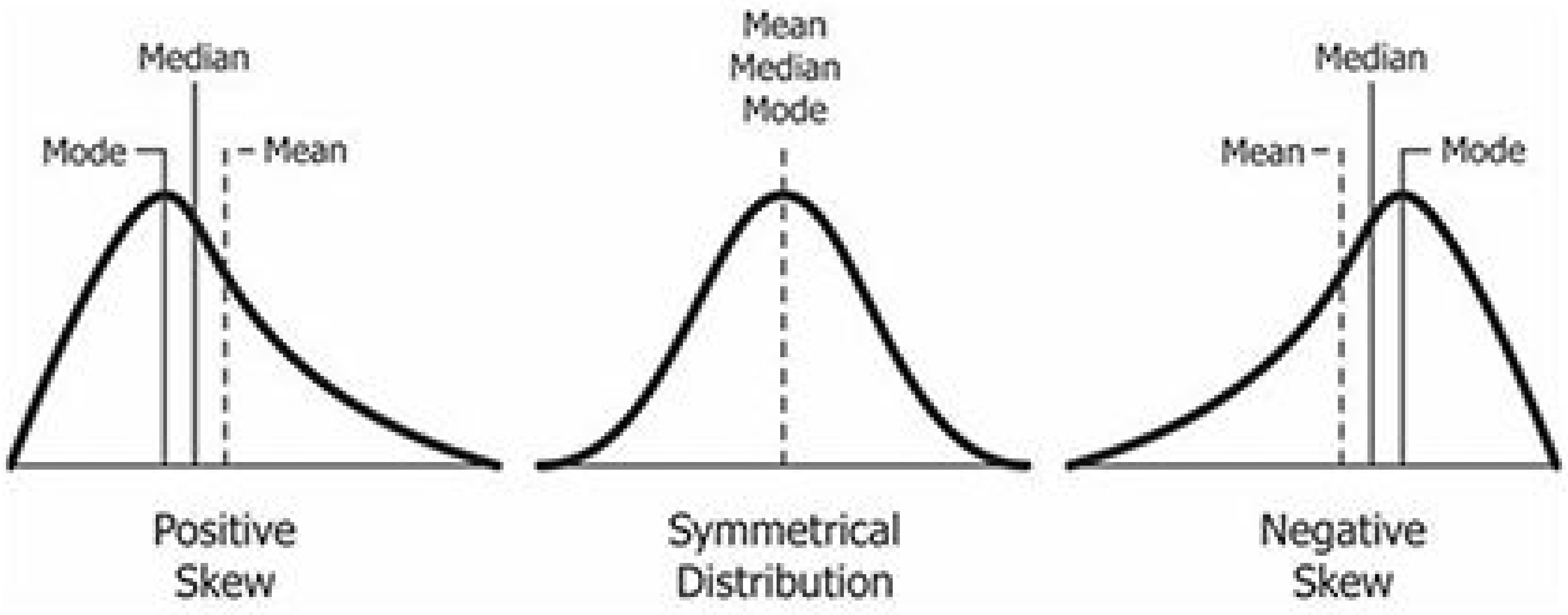
MEAN VS MEDIAN

MEAN

The mean is very susceptible to the influence of outliers so that the mean value is not good enough to describe the center of the data for asymmetric data

MEDIAN

Because the median value is not determined by calculation but rather by the position of the data, outliers do not have a large impact on the median value.



MISSING VALUES

- Missing values is a condition where a variable has no value
- In dataframes, missing values are usually marked with NaN

	Part	Feature	Price
	Monitor	LED	12,000
1	CPU	i7	35000
2	NaN	RGB	NaN
3	Mouse	Wireless	1200
4	NaN	Zebronics	NaN
5	Extentions	NaN	250
6	Table	Urban clap	7000
7	Chair	Apex Chairs	12000
8	Wifi Connection	Airtel	799

Missing values are marked with NaN

TYPES OF MISSING VALUES

1

MISSING COMPLETELY AT RANDOM

2

MISSING AT RANDOM

3

MISSING NOT AT RANDOM

TYPES OF MISSING VALUES: MCAR

MCAR (missing completely at random) is a condition where missing values occur randomly. The occurrence of missing values has no relationship with other data. The causes of this type include: human/system error.

TYPES OF MISSING VALUES: MAR

MAR (missing at random) is a condition where there is a relationship between the occurrence of missing values with other variables or in other words the pattern of occurrence of missing values can be explained from other variables.

TYPES OF MISSING VALUES: MNAR

MNAR (missing not at random) is a condition where the pattern of missing values appears cannot be explained by other variable. This type usually occursbecause of the person's reluctance to provide answers or side bias.

WHY HANDLING MISSING VALUES IS IMPORTANT?

- Some algorithm cannot be used if the dataset contains missing values
- Produce biased analysis result
- Reduce statistics analysis accuracy

HOW TO FIND OUT THE EXISTENCE OF MISSING VALUES

```
df = pd.read_csv('titanic.csv')
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

There is data titanic in above

HOW TO FIND OUT THE EXISTENCE OF MISSING VALUES

```
df.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

count the number of missing value in every column

```
df.isnull().sum() / len(df) * 100.0
```

```
PassengerId     0.000000
Survived        0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age            19.865320
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.000000
Cabin          77.104377
Embarked        0.224467
dtype: float64
```

count the number of missing value in persent

SHOW DATA WITH MISSING VALUES

df [df [< column_name >] . isnull () == True]

```
df[df['Age'].isnull() == True]
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
17	18	1	2	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	13.0000	NaN	S
19	20	1	3	Masselmani, Mrs. Fatima	female	NaN	0	0	2649	7.2250	NaN	C
26	27	0	3	Emir, Mr. Farred Chehab	male	NaN	0	0	2631	7.2250	NaN	C
28	29	1	3	O'Dwyer, Miss. Ellen "Nellie"	female	NaN	0	0	330959	7.8792	NaN	Q
...	...	...	...	...	...	...	...	...	...	...	...	...
859	860	0	3	Razi, Mr. Raihed	male	NaN	0	0	2629	7.2292	NaN	C
863	864	0	3	Sage, Miss. Dorothy Edith "Dolly"	female	NaN	8	2	CA. 2343	69.5500	NaN	S
868	869	0	3	van Melkebeke, Mr. Philemon	male	NaN	0	0	345777	9.5000	NaN	S
878	879	0	3	Laleff, Mr. Kristo	male	NaN	0	0	349217	7.8958	NaN	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S

177 rows × 12 columns

show all the rows with missing values in column Age

HANDLING MISSING VALUES

The strategy for handling with missing values must be based on understand the reasons behind the appearance of missing values. There are 2 strategies for dealing with missing values:

- Delesi (delete)
- Imputation (filled with a value)

DELESI (DELETE)

This method is not recommended especially for the MNAR type. The disadvantage of this method is the possibility of data the important thing is also deleted. Deletion can be done in two ways:

- Row deletion `df.dropna(axis=0)`
- Column deletion `df.dropna(axis=1)`

`dropna` will remove rows/columns containing missing values

DELESI COLUMN

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
```

BEFORE

```
delete_row = df.dropna(axis=0)
delete_row.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 183 entries, 1 to 889
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  183 non-null    int64  
 1   Survived     183 non-null    int64  
 2   Pclass       183 non-null    int64  
 3   Name         183 non-null    object 
 4   Sex          183 non-null    object 
 5   Age          183 non-null    float64
 6   SibSp        183 non-null    int64  
 7   Parch        183 non-null    int64  
 8   Ticket       183 non-null    object 
 9   Fare         183 non-null    float64
 10  Cabin        183 non-null    object 
 11  Embarked     183 non-null    object 
dtypes: float64(2), int64(5), object(5)
```

AFTER

DELESI COLUMN

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
```

BEFORE

```
delete_col = df.dropna(axis=1)
delete_col.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   SibSp        891 non-null    int64  
 6   Parch        891 non-null    int64  
 7   Ticket       891 non-null    object 
 8   Fare         891 non-null    float64
dtypes: float64(1), int64(5), object(3)
```

AFTER

IMPUTATION

Imputation is the process of filling in missing values with a value. In pandas, imputation can be done using fillna() `fillna()`

```
df[ < column_name > ] . fillna( < value > )
```

`< value >` can be replaced with a scalar value or a calculation result such as mean, min, max, or mode. Additionally, missing values can be filled with valid values before or after (backfill vs frontfill)

IMPUTATION

```
df['Age'].fillna(20)
```

```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888    20.0
889    26.0
890    32.0
```

FILL WITH SCALAR VALUE

```
df['Age'].fillna(df['Age'].mean())
```

```
0      22.000000
1      38.000000
2      26.000000
3      35.000000
4      35.000000
         ...
886    27.000000
887    19.000000
888    29.699118
889    26.000000
890    32.000000
```

FILL WITH AGGREGATE FUNCTION VALUE

```
df['Embarked'].fillna(df['Embarked'].mode()[0])
```

```
0      S
1      C
2      S
3      S
4      S
      ..
886    S
887    S
888    S
889    C
890    Q
```

FILL WITH MODE VALUE

BACKFILL VS FRONTFILL

Make a new dataframe that contains missing values

```python
df = pd.DataFrame([('Foreign Cinema', 'Resturant', pd.NA),
                   ('Liho liho', 'Restaurant', 224.0),
                   (pd.NA, 'bar', 80.5),
                   (pd.NA, 'bar', pd.NA),
                   (pd.NA, 'bar', 65.23),
                   ('Blue Barn', pd.NA, 361.98)],
          columns=('name', 'type', 'AvgBill')
                   )
```

```python
df
```

	name	type	AvgBill
0	Foreign Cinema	Resturant	<NA>
1	Liho liho	Restaurant	224.0
2	<NA>	bar	80.5
3	<NA>	bar	<NA>
4	<NA>	bar	65.23
5	Blue Barn	<NA>	361.98

BEFORE

```python
df['name'].fillna(method='bfill')
```

```
0    Foreign Cinema
1         Liho liho
2         Blue Barn
3         Blue Barn
4         Blue Barn
5         Blue Barn
Name: name, dtype: object
```

BACKFILL

```python
df['name'].fillna(method='ffill')
```

```
0    Foreign Cinema
1         Liho liho
2         Liho liho
3         Liho liho
4         Liho liho
5         Blue Barn
Name: name, dtype: object
```

FRONTFILL

THANK YOU

Linkedin:
https://www.linkedin.com/in/anitamilaoktafani/