EXPLORATORY DATA SCIENCE

#DigitalSkillFair39

By : Fadhil Haidar Rais

ibimbing

IMPORT LIBRARY & LOAD DATA

The following code snippet successfully imports and loads the Titanic passenger dataset into memory using the pandas library. This dataset is well-suited for various types of data exploration and analysis, including :

- Examining survival patterns based on gender, age, or passenger class.
- Analyzing ticket fare distributions and embarkation points.
- Studying family relationships through sibling/spouse and parent/child data.

```
import pandas as pd

data = pd.read_csv('/content/Titanic-Dataset.csv')
```

```
data
```

	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
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

CHECKING MISSING VALUE

The data.info() function provides a concise summary of the Titanic dataset's structure. It includes the number of entries, column names, data types, and the count of non-null values in each column.

Some columns have missing values:

- Age: 714 non-null (177 missing)
- Cabin: only 204 non-null
- Embarked: 2 missing

Data types:

- int64 for numbers (e.g., PassengerId, Pclass)
- float64 for Age, Fare
- object for text (e.g., Name, Sex, Ticket)

```
[ ] data.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)
memory usage: 83.7+ KB
```

CHECKING MISSING VALUE

The code data.isna().sum() checks for missing values (NaN) in each column of the dataset data. It counts and displays how many missing entries exist per column, helping identify which data requires cleaning or imputation.

Key Points:

- Function Purpose: The code detects missing values in a dataset by combining isna() (identifies NaNs) and sum() (counts them per column).
- Output Meaning: Columns like Age (177 missing), Cabin (687 missing), and Embarked (2 missing) show where data is incomplete.
- Common Actions:
  - Impute missing Age values (e.g., with median).
  - Drop or ignore Cabin due to excessive missingness.
  - Fill Embarked's 2 missing values with the mode.

```
data.isna().sum()
```

	0
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

RESOLVING MISSING VALUE

The provided code snippet handles missing values (NaN) in a dataset called data by filling them based on the column's data type. This ensures the dataset is complete and ready for analysis or modeling.

```
# Mengatasi missing value
for column in data.columns:
    if data[column].dtype == 'object':
        # Jika kolom bertipe object, isi dengan mode
        data[column].fillna(data[column].mode()[0], inplace=True)
    else:
        # Jika kolom bertipe numerik, isi dengan mean
        data[column].fillna(data[column].mean(), inplace=True)
```

- Column Iteration:
  - The code loops through every column in the DataFrame using for column in data.columns
- Data Type Handling:
  - For categorical columns (dtype='object'), it fills missing values with the mode (most frequent value) using mode()[0]
  - For numerical columns, it uses the mean value to fill gaps via mean()
- In-Place Modification:
  - The inplace=True parameter ensures all changes are applied directly to the original DataFrame
  - This avoids the need to create a new DataFrame object

Checking and Resolving Duplicate Data

This code checks for, removes, and verifies duplicate data entries, ensuring a clean dataset through systematic duplicate detection and removal.

Key Points:

- Duplicate Detection:
  - Uses data.duplicated().sum() to count duplicate rows across all columns.
  - Output shows zero duplicates (Jumlah data yang duplikat = 0).
- Duplicate Removal:
  - Executes data.drop_duplicates() to clean the dataset, even if no duplicates exist.
  - Acts as a safeguard to maintain data integrity.
- Result Verification:
  - Rechecks for duplicates after removal to confirm successful cleaning.
  - Final output confirms no duplicates remain (Jumlah data yang duplikat = 0).

```
[ ] # Mengecek apakah ada duplicate di seluruh kolom
    check_duplicate = data.duplicated().sum()

    print(f"Jumlah data yang duplikat = {check_duplicate}")

    Jumlah data yang duplikat = 0

[ ] # Handling duplicate
    data = data.drop_duplicates()

[ ] # Mengecek duplicate setelah di-handle
    handle_duplicate = data.duplicated().sum()

    print(f"Jumlah data yang duplikat = {handle_duplicate}")

    Jumlah data yang duplikat = 0
```

THANK YOU

Do you have any question?

+123-456-7890
hello@reallygreatsite.com
www.reallygreatsite.com